Bloomberg Businessweek

Technology

http://www.businessweek.com/articles/2013-10-16/open-source-everything-the-moral-of-the-healthcare-dot-gov-debacle

The Obamacare Website Didn't Have to Fail. How to Do Better Next Time

By Paul Ford October 16, 2013

Right as the U.S. government shut down on Oct. 1, the new national health-care exchange went live at healthcare.gov. Intended for people who want to buy health insurance under the provisions of the Affordable Care Act, the website failed from the start. It was unusable to hundreds of thousands of Americans who wanted to enroll. Screens wouldn't load. A woman trying to get help via Web chat was told to "please be patient" an infuriating 40 times. Despite pledges from Obama administration officials that the "glitches" would be quickly fixed, the list of bugs and problems kept growing.

Healthcare.gov isn't just a website; it's more like a platform for building health-care marketplaces. Visiting the site is like visiting a restaurant. You sit in the dining room, read the menu, and tell the waiter what you want, and off he goes to the kitchen with your order. The dining room is the front end, with all the buttons to click and forms to fill out. The kitchen is the back end, with all the databases and services. The contractor most responsible for the back end is CGI Federal. Apparently it's this company's part of the system that's burning up under the load of thousands of simultaneous users.

But it's not just that the kitchen's on fire. The alarm system doesn't work either. The waiters can't hear the screams of the cooks. It's hard for outsiders to tell exactly what's going on, because CGI Federal's code is locked away out of sight, and the company hasn't replied to press inquiries (including those from *Bloomberg Businessweek*). You can't get a waiter's attention to save your life.

Put charitably, the rollout of healthcare.gov has been a mess. Millward Brown Digital, a consulting firm, reports that a mere 1 percent of the 3.7 million people who tried to register on the federal exchange in the first week actually managed to enroll. Even if the problems are fixed, the debacle makes clear that it's time for the government to change the way it ships code—namely, by embracing the approach to software development that has revolutionized the technology industry.

Companies such as Google (GOOG), Amazon.com (AMZN), Twitter, and Facebook (FB) all think in terms of platforms talking to applications. They deploy lots of small teams that are expected to ship new features and fixes all the time—sometimes daily. Like anything that involves human beings, shipping code can devolve into squabbling, missed deadlines, and flawed releases. The programming community's key realization is that the solution to these problems is to create more transparency, not less: code reviews, tons of "unit tests" to automatically find flaws, scheduled stand-up meetings, and the constant pushing of new code into the open, where it's used by real people. To cite just one example, developers at the giant online marketplace Etsy are encouraged to release code to the world on their first day of work.

Government IT can't work in such a transparent way. Or could it? There's a whole set of tools, methods, and processes already set up and ready to use, all embodied in the culture of open-source software development. The U.S. federal government, led by the executive branch, should make all taxpayer-funded software

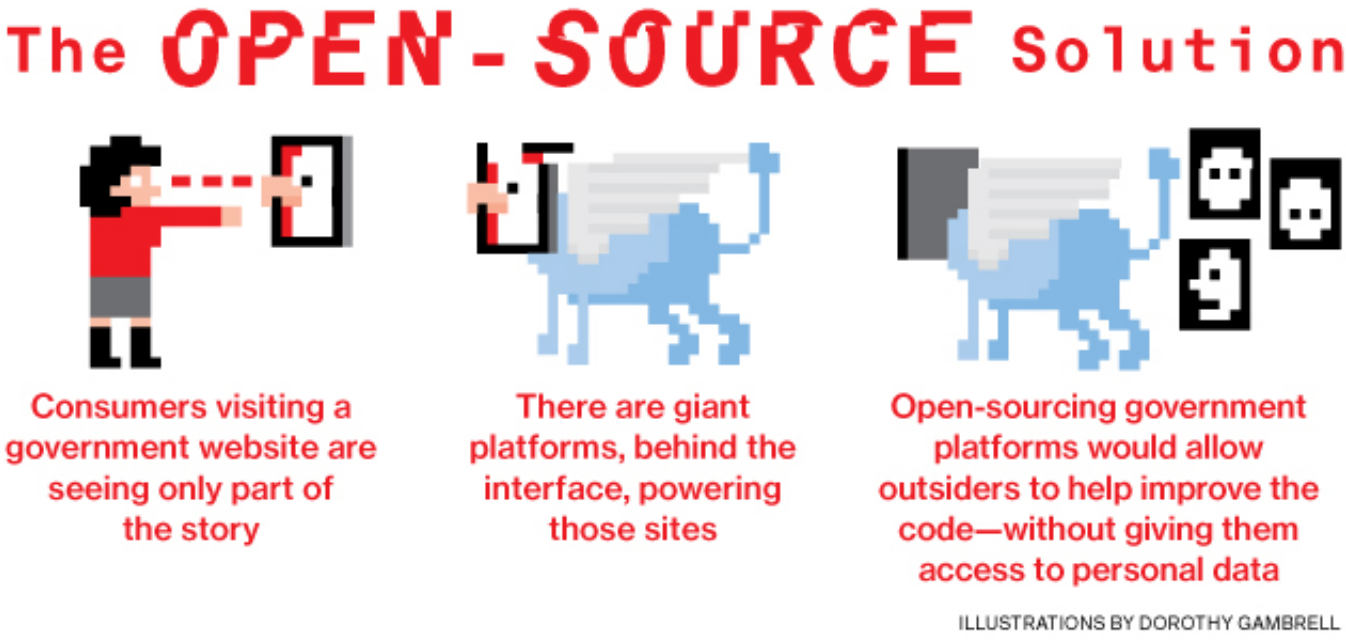
development open-sourced by default. In the short run, this would help to prevent the recurrence of problems like those that plague healthcare.gov. Longer term, it will lead to better, more secure software and could allow the government to deliver a range of services more effectively. And it would enrich democracy to boot.

The basic goal of the free software movement is to make useful software code available to anyone who wants it. Thirty years ago this sounded like communism, because code was seen as a kind of property. But in recent decades many people have come to believe that software code is more like a conversation. (As one famous programming textbook put it, "Programs must be written for people to read, and only incidentally for machines to execute.") That's why people say that free software is free as in free speech, not as in beer.

Want to open-source code? Choose a free software license and release your code online with the text of that license attached. That's all it takes. History shows, however, that just licensing code and making it available isn't enough. You need to create a culture around your project and engage with other people doing related work. If you do a good job of it, you and your collaborators can create great, first-class, highly secure software. Web browsers such as Mozilla Firefox and Google Chrome were built this way. It's very likely that the smartphone in your pocket is built on top of an open-source kernel.

The government has an advantage over typical open-source projects. People, including programmers, are intrinsically interested in what it's doing, often because their lives are affected directly. If it wanted to, the U.S. could tap an army of interested coders ready to support official efforts.

Interestingly, the first released version of healthcare.gov was open-sourced. When it was launched in June, the initial Web front end (the dining room, not the kitchen) was widely lauded as a modern, well-developed website—pretty good for government work. It was coded by a company called Development Seed, a Washington (D.C.)-based startup. All of its work on healthcare.gov was made available on GitHub, a massive open-source collaboration platform.



Development Seed was brought in as a subcontractor specifically because it had lots of code on GitHub. Its part of the healthcare.gov project was "open by default," says Eric Gundersen, Development Seed's president. When it launched, the company "open-sourced the whole repo" (shorthand for "code repository"). It was built atop open frameworks such as Bootstrap, a spinoff from Twitter, and Backbone.js, a spinoff from the nonprofit DocumentCloud. Using these solutions saved countless hours and cost nothing. Along the way,

says Gundersen, Development Seed contributed improvements to the software it was repurposing. This approach—take a lot, give a little—is what makes open source work; all of the little changes add up across millions of projects. "From start to finish of the entire project, it was three months and 25 days," says Gundersen. "We were operating in a very fast sprint cycle." (The whole open repository for healthcare.gov has since been removed from GitHub without a clear explanation, although third parties have been able to restore the code from external backups.)

The engineers of Development Seed are not heroes of a revolution. They are government subcontractors who did their assigned job using best practices in a transparent manner. Gundersen himself points out that there is nothing inherently unusual about the methods his firm uses. The end result was a front end that worked, even if the rest of healthcare.gov didn't. "We were disappointed with the launch because we wanted to buy health care for our team," Gundersen says.

If you've never looked at GitHub, take some time to explore the site. It's built on top of Git, an open-source "revision control" system that provides a way to track changes to projects over time. You can fast-forward or rewind through history to see how projects grow and evolve, and create derivative projects with one command. You can also file bugs or send in changes, which project managers can accept or reject. Some of the projects go back decades.

The initial release of healthcare.gov was built along these principles, but the real core of the site—the much larger, more complex back end—wasn't. The process remains opaque. According to the Sunlight Foundation, a nonpartisan group focused on making government more accountable, at least 47 contractors have been involved with the Affordable Care Act in some capacity. The White House has yet to release its own explanation of what happened. So we don't know what went wrong; we don't know how it was built; and as far as anyone can tell, the project didn't give anything back to the larger culture of code development.

Why isn't an open, proven process, like the one that GitHub and other sites make available for free, the norm for government work? Commenting on the failure of healthcare.gov, Clay Johnson of the Department of Better Technology, a company that designs and builds software for government, writes that "6,500 pages of regulation, cumbersome business registration processes, and hostile bidding environments ensure that very few new businesses can compete for contracts." Plus, in the era of WikiLeaks and Edward Snowden, the fear of security breaches makes bureaucrats allergic to a transparent software-development process. According to Tom Lee, director of Sunlight Labs, "the single biggest obstacle is that opening data is seen as all downside risk."

Software systems are so large now that they can no longer exist separately from other systems. Code always depends on other code. It's never finished: Write a piece of software today, and you're likely to be debugging it a decade later. Modern software development is more like sustainable forestry, where you expect to keep coming back to the same grove year after year. The technology industry has come to understand that software is a process. The U.S. government, however, keeps insisting it's a product—ironic, given that we can trace our history back to an open Constitution that defined, in Article V, the means of amendment.

Other countries already have national mandates around open source, says Deborah Bryant, an expert in the public-sector adoption of open-source software and methodologies. She lists Brazil, Spain, France, and Malaysia as examples. "I think that the White House developing an executive order in the area of open source would be extremely helpful. For every [civil servant] who has gone forward on doing something with open source, there are probably another dozen who would like to have that kind of cover" that an executive order would provide.

What about security? If everyone can see the code, won't hackers have a way in? No. First of all, no one's arguing that private user data be made open—just the code that handles that data. The U.S. Department of

Defense, an organization with some fairly stringent security concerns, has a great FAQ on open source. "Does the DoD use [open-source software] for security functions?" is one of the questions. The answer is an unambiguous "yes." Getting rid of open-source code used for defense "would have immediate, broad, and in some cases strongly negative impacts on the ability of the DoD to analyze and protect its own networks against hostile intrusion."

Right now, hundreds of thousands of people use open-source software when developing projects. That includes the world's largest corporations, countless startups, and many government agencies. There are scores of public-sector open-source projects (many of which are unavailable during the shutdown). The groundwork is there for a major initiative to make open source the standard for government work. All that said, open sourcing isn't a panacea, as Tom Lee cautions. "The exact problems with healthcare.gov aren't yet fully known," he says. "And open processes can sometimes be slower. They also don't always attract constructive attention unless they're genuinely useful."

He's right. Before it was pulled down, the healthcare.gov code that Development Seed posted on GitHub had a bug report complaining about "too many death panels." But there are genuine bug reports, too, and plenty of people who would like to help and are ready to be more engaged with the government through bug reports and code patches. It may sound odd, but some people will file bug reports and track down problems for the joy of it. Nurturing this openness helps everyone and brings people into a new, weird, and exciting civic process.

When asked for his thoughts on healthcare.gov, Victor Lombardi, author of *Why We Fail: Learning from Experience Design Failures*, was optimistic. "The problems don't sound catastrophic," he wrote via e-mail. "I think the media just needs something to talk about." (Cough.) "History may judge this project as the catalyst that revolutionized the United States health-care system," he concluded, "and no one will remember a few hiccups at launch."

> We are witnessing the start of the most hellish code review in software development history

Doubtless the problems will get fixed. All bugs are shallow with the president watching. In the meantime, it's clear that tens of millions of dollars have been spent to launch something broken. (The actual numbers are disturbingly hard to pin down, another place where transparency would help.) And the Republican leadership has wasted no time making hay: Senator Lamar Alexander and Representative Darrell Issa sent a letter to Kathleen Sebelius, secretary of Health and Human Services, demanding answers to a list of strongly worded, highly technical—and fair—questions. Speaker of the House John Boehner has called the rollout a "train wreck"; Senator Pat Roberts has called for Sebelius to resign. We are witnessing the beginnings of the single most exhausting, hellish code review in software development history.

Regardless of your opinions on the health-care law, this is the wrong way to make software. We may never be able to stop websites from crashing or keep bugs from creeping into code. But opening the process would expose what went wrong and why. People can still be paid large amounts of money to code for the government, if that's what it takes. They just shouldn't do it in secret. The U.S. needs to say that all development will take place in the open, using a tool like GitHub, or GitHub itself. If some things can't be open, for reasons of security or copyright, fine; say so and get an exemption.

Fixing what ails healthcare.gov will be costly and politically contentious. But it's also an opportunity to learn why things went wrong and make sure these problems don't repeat. With more and more government services delivered as software over the Internet, it's imperative that the code behind it be made public. An open society should be an open-source society.

Ford is a programmer and the creator of SavePublishing.com.