6 Software Development Lessons From Healthcare.gov's Failed Launch

The problems that plagued the launch of Healthcare.gov -- the online data hub and insurance marketplace central to healthcare reform -- will someday fill a book. For now, developers can apply these six lessons from Healthcare.gov's very public failure to their own software projects.

"I spent $174 million on a website and all I got was this bad press."

— Someone, somewhere in the U.S. Department of Health and Human Services (HHS)

Well, at least someone should have said it.

Healthcare.gov is arguably the most public software failure of the decade. You may have read commentary by people who have never had to write or test code, never served on a software project, and likely don't know how to right-click and "view source" and read the HTML. That's OK; most journalists, most of the time, don't need to be very technical.

Tutorial: Healthcare.gov's Problems: What We Know So Far

More: Why is the Obamacare Website So Sick?

This article tries to go further than the typical coverage of Healthcare.gov. The amazing thing about this story isn't the failure. That was fairly obvious. No, the strange thing is the manner in which often conflicting information is coming out. Writing this piece requires some archeology: Going over facts and looking for inconsistencies to assemble the best information about what's happened and pinpoint six lessons we might learn from it.
Healthcare.gov was a great idea. If only it worked …

1. Sprints and Iterations Do Not an 'Agile' Project Make

One of the biggest arguments about the Healthcare.gov debacle is the development method. Pundits suggest that, since war room notes use terms such as sprints and story cards, the project used an agile approach, but agile approaches can not prevent all risk. Even the reduction of risks requires skill and judgement. Then again, CBS News claims that Healthcare.gov shrunk its schedule for testing from months to weeks, suggesting that Healthcare.gov was never properly tested.

In agile software development, the terms "sprint" and "iteration" mean the time for a new, completed chunk of software development to be designed, coded and fully tested, end-to-end. The standard length for teams to start with iterations is two weeks; improvement beyond that generally means shortening the iteration.

How-to: 3 Ways to Be More Agile With Software Shipping Decisions

More: Can New Software Testing Frameworks Bring Us to Provably Correct Software?

Many teams use "sprint" or "iteration," only to insert waterfall concepts. Language such as "three architecture sprints, six coding sprints, two test sprints and two hardening sprints" is usually a clue that something's wrong.
But there's something else more sinister in the HealthCare.gov project.

2. The System Produced the Outcome, Not the Lack of Testing

If you've worked on a waterfall project with a defined test phase, you know it's not really a "test" phase at all. Once the first show-stopper bug is found, it's actually a fixing phase.

To state that Healthcare.gov wasn't fully tested implies that the test group either never found any show-stopper bugs or didn't have the ability to halt the project. Given the legally required go-live mandate, the second option seems realistic.

How-to: How to Adjust to the Changing Face of Software Testing

More: How to Deal With Software Development Schedule Pressure

One thing we do know for sure: The organization rushed ahead with coding before it knew answers to questions. That's no recipe for success.

We know because at go-live the JavaScript code itself was still full of filler text that a non-decision maker typically insert onto a page or pop-up. Programmers know that something goes wrong but need to get the language approved, so they write *lorem ipsum* and move on. This is exactly the kind of thing that happens when people are under a deadline and can't get answers to their questions.

Mike Adams, editor of NaturalNews.com, found dozens of critical JavaScript errors by simply viewing the source and following links. Here's a small sample:

"TODO: add functionality to show alert text after too many tries at log in"

"make sure we don't try to do this before the saml has been posted if (window.registrationInitialSessionCallsComplete)"

"Attention: This file is generated once and can be modified by hand"
People in the organization must have known the site was not ready. Perhaps they spoke up and were overruled. We don’t know.

The problem at Healthcare.gov wasn’t a lack of testing. It was a lack of critical thinking. Still, the site did have a prime testing contractor. Sure, the testers didn’t have months, but they did have weeks. What were they doing?

3. Testing Should Be Part of the Delivery Process

QSSI was the lead test contractor on Healthcare.gov, providing services it refers to as Independent Verification and Validation, or IV&V.

Here’s what QSSI has to say about its own IV&V Process (links added):

Our IV&V services are consistent with the latest systems engineering and process improvement models, and are derived from industry standards, including the IEEE Standard 1012-2004 Standard for Software Verification and Validation and the CMMI process maturity framework. QSSI provides an objective assessment of products and processes throughout the project life cycle in an environment free from the influence, guidance, and control of the development effort. Services include: criticality analysis, requirements analysis and tracing, software design analysis, milestone reviews and metrics, code review and analysis, document inspection [and] defect investigation, plus training evaluation, planning, execution, reporting and witnessing.

Look at the QSSI careers site and you see a lot of focus on these standards, policies and procedures — in particular, that IV&V involves tracing requirements to implementation.

That’s one aspect of product risk, but it’s certainly not a whole-process look. The agile manifesto changed the focus of software from “processes and tools” to “individuals and interactions” back in 2001. That same year, Joel Spolsky suggested it was possible to pass every functional test yet still have a product that isn’t usable.

The real reveal, however, comes from the 175 pages of war room notes from the early release of the website. The notes focus entirely on dealing with trouble tickets that occur in the field. There’s no connection to testing. At no point in the document — nowhere — does anyone go back and say, “Here’s the list of things deferred at go-live as known issues.”

The entire process is firefighting. Testing wasn’t seriously considered. If testing actually occurred — and real testing, not fake testing — it failed to make any meaningful connection to operations or support or project management. Testing failed to recommend a no-go decision.


More: Obamacare Exchange Contractors had Past Security Lapses

If testing can’t impact the schedule or product in any way, it’s just waste. Better to eliminate it altogether.
4. Do It Manually Before You Automate

After a great deal of failure and public fallout from Healthcare.gov, customers started to use the paper application process as an alternative. Sounds reasonable — until you realize the paper applications would be mailed to a customer service center using the same Web-based technology to determine eligibility.

In other words: If you give up on Healthcare.gov because you can't purchase insurance, you can mail a paper application to a processing center where a human will try to enroll you manually in Healthcare.gov. That doesn't work.

News: U.S. CIO Sees Healthcare.gov Glitches as a 'Teachable Moment'

More: Officials Target End of November for Smooth Healthcare.gov

Neither the federal government nor any known contractor has any way of ensuring eligibility manually. If they had an 800-number to the IRS, and another to the Department of Homeland Security, the agencies could hire an army of temps to manage the order processing. (That idea's not that farfetched; it's essentially how the federal government conducts the census every 10 years.)

If you have a large-scale adoption and have to flip the switch, make sure you have a way to flip it back and can execute manually.

5. The System Had to Be Perfect, By Law, But It Wasn't

Healthcare.gov must verify a would-be applicant's eligibility from Homeland Security, IRS and Social Security systems, in real time, plus enroll members electronically to any registered insurance company in any of the 34 states that don't have their own state health insurance exchange.

Information needs to be completely accurate, so every request needs to run-submit information, over Web services, to an insurance company. That means users click Submit, then wait for a back-end Web services call, and wait and wait and …

News: Healthcare.gov Team Improves Site Response Times

Officials: We Warned of Security Risks Just Before Healthcare.gov Went Live

All this because the system had to work in "real time," or synchronously. Except it didn't.

Turns out that HHS created a spreadsheet with the basic rate table information. Healthcare.gov could have been a very simple process, then: Type in zip code, click Submit, select age and family factors, get quote and get number to call purchase insurance.

We know this smaller scope could have worked because it does work. A tiny company called Opscost took that spreadsheet and implemented a website doing just that in few than two person-weeks. The site is thehealthsherpa.com (shown below).
This comparison of six person-days to hundreds of millions of dollars is obviously unfair. HealthCare.gov had to do more, including end-to-end enrollment. Healthcare.gov customers were supposed to be able to sign up on a government Webform and have the transaction automatically flow into an insurer’s system.

Instead of an 80 percent system for a fraction of the cost, the customer insisted on 100 percent by force of law. Lacking the capability to actually develop such a system, the contractors made their best attempts and shipped whatever they had on the end date. The results, while tragic, aren’t exactly a surprise.

George Kalogeropoulos, customer support at OpsCost, says the company’s biggest surprise while validating the market was that a huge number of customers wanted to window shop. They wanted a quote without giving up personal information such as phone number or Social Security Number.

**Analysis: Why Healthcare.gov May Be a 'Black Swan'**

**More: Lawmakers Seek Answers on Obamacare Data Hub Security**

At go-live, Healthcare.gov worked on the opposite model, forcing personal information before allowing users to see a quote. (This policy has since changed.) That allowed Healthcare.gov to force users to agree to terms of service, and to connect to backend servers at the IRS, Homeland Security and other agencies. Those are great features for the end of the process, but when your users want to window shop, perhaps less so.

**6: Threat Modeling Matters**
Ben Simo isn’t a black-hat hacker. He doesn’t spend his time trying to "crack" things. He’s a developer/tester who knows how to view source code and use the developer tab to look at back-end Web transmissions, mostly RESTful services. That’s exactly what Simo did with Healthcare.gov: Look at how his family’s information was transmitted as he tried to sign up.

Simo didn’t succeed in signing up — but he did find a site that transmitted his username and email in plain text, loaded insecure resources and could reveal user IDs and email addresses through login notification messages.

Perhaps worst of all, the password reset feature results in data combinations that could enable phishing attacks. An attacker logged in as you can get personal information from REST service responses, including name, address, date of birth and Social Security number. That's exactly the kind of information identity thieves can use to get false credit cards or gain access to bank accounts.

Related: Healthcare.gov Granted a Waiver to Launch Despite High Risk Levels

More: Scramble to Fix Healthcare.gov Site Heightens Security Risks

It appears that Ben Simo was more concerned about personal information appearing on the Web than the people who made Healthcare.gov.

Simo’s analysis has been quoted in TIME, on CNN and during HHS Secretary Kathleen Sebelius’ Congressional testimony. Simo is a former president of the Association for Software Testing, and I’ve served on its board of directors. We’re regular humans. How did one of us find not a single defect but a systematic mess?

It’s likely that security testing was faked, overlooked or ignored for Healthcare.gov, just like functional testing. Don’t let this happen on your watch.

Ultimate Lesson from Healthcare.gov: Take Incremental Approach
It's early in the Healthcare.gov lifecycle. We know the project's failure was systemic — multiple failures on multiple levels — but we don't know exactly what did happen. Identifying all the issues might make for a good book someday, but it's far too much for one article.

Putting together the six lessons here, though, and you'll see a pattern: Healthcare.gov was a single, Big Bang rollout that couldn't be stopped.

At this early stage, if you can take only one thing from Healthcare.gov, it's this: Use incremental approaches such as betas, early testing and regular delivery of a completely tested system, in tandem with flexible scope to find risk before it finds you.

The saddest part about the Healthcare.gov failure isn't that is so massive. The saddest part is that the failure was both predictable and greatly preventable.

Matthew Heusser is a consultant and writer based in West Michigan. You can follow Matt on Twitter @mheusser, contact him by email or visit the website of his company, Excelon Development. Follow everything from CIO.com on Twitter @CIOonline, Facebook, Google + and LinkedIn.

Matthew Heusser
Matthew Heusser is principal consultant of Excelon Development. He is based in West Michigan.
Programmers In Israel Are Puzzled By This Link
TBHR

You Have Giraffes in Your Data!
Optimove Blog

The Most Exciting MMORPG You've Ever Played. Don't miss t...
Sparta Online Game

Build A Beautiful Simple Website For Your Small Business
Wix.com

This cloud helps Special Olympics track data for 4.8 million a...

6 Things Galaxy S6 edge+ does that iPhone 6s Plus can't

Windows 10 review: Hold off if you use Windows 7

Does Volatility Signal The End of the Bull Market?
The Financialist by Credit Suisse