Failed case : HealthCare.gov

Team Blue Mondays

Contents

Initial complexity

The construction of HealthCare.gov was overseen by Centers for Medicare and Medicaid Services (CMS) which is a part of the US Department of Health and Human Services (HHS). The primary contractor was CGI who had a contract of around 250 million dollars. Next to that there were 16 official subcontractors, but the total number of subcontractors was actually 55. All under the supervison of CMS. There were also 300 private insurers. All in all there were around 4000 plans.

The clients were according to [4, page 6]: the Department of Health and Human Services (HHS), Centers for Medicare & Medicaid Services (CMS), 36 States, 300 private insurers, U.S. Chief Technology Office, GAO, Media, Citizens, Social Security Administration, the Internal Revenue Service, Veterans Administration, Office of Personnel Management, Peace Corps, etc.

The internal architecture of HealthCare.gov is very complex. US residents who want to apply for an insurance needed a lot of information from the different systems. Like their income and immigration status. The website needed to be connected with all the different systems from the federal government and the private insurers, like the Internal Revenue Service, Social Security Administration, the Peace Corps, etc. The idea was the information of all the different federal systems should be retrieved real time. According to [4] the U.S. Chief Technology Officer Todd Park has said that the government expected HealthCare.gov to draw 50,000 to 60,000 simultaneous users. But in the first week the site was overwhelmed by up to five times as many users.

Timeline of problems and solutions

Before the launch

There were a lot of warnings before the lauch of the website by different parties. But is seems nothing was done with them. Below are a couple of them noted [4].

On April 4, 2013 a 15-page document by McKinsey & Co. warns among others the CMS administrator that there was insufficient end-to-end testing and advises that a limited initial launch of the website would be ideal. It seems nothing is done with this report, because it surfaced only half November 2013. One and a half month after the lauch. Next to that there were only two weeks of end-to-end testing before the launch took place for all the users.

The Government Accountability Office (GAO), an independent organisation of the government, reported in June 2013 that there could be trouble with the goal of launching the website on the first of October 2013.

On 10 September there was a panel of the House of Representatives who were hearing among others consulting firm Leavitt Partners and contractor CGI. CGI stated during this hearing that it was on schedule for launching the website. On the other hand Leavitt Partners stated that the launch will be 'rocky' [4, page 7] because there are still ongoing technology challenges for the online health insurance exchange.

According to the New York Times [4, page 7], documents released by House investigators reveal that the 'testing bulletin' showed that the website as of September 30 could handle only about 1100 users at a time, even though officials have said it should have been abble to accomodate perhaps as many as 60,000 users.

After the launch [4]

At its launch (1 October 2013) the website did not perform well. Only 271,000 of the 9.47 million users that try to registrate where succesfull. ?There was already $700 million dollar spend? In 2014 the total number of costs of the project is 1.7 billion dollar (reference). Suddenly, one question arises. What would have happened if they had established a schedule of access to the site?. There were official warnings telling that the site was going to be unable to accommodate so many users at once.

An extra team is recluted (22th October 2013) It was recognized that product testing was insufficient. Serious underestimation of time.

Quality Software Services, Inc. (QSSI) is selected as the contractor responsible to now oversee federal website fixes (October 27, 2013). The *failure reason* points to a technical complexity (a single data center brings down the most critical mission, 'the website' this should be presumed and avoided). The *failure factor* leans at requirements that should be identi?ed and addressed before project completion.

(October 31, 2013) Again an unrealistic schedule. Site crashed for second time due small amount of servers, an explicit gap at project definition and planning.

(November 14, 2013) President Obama declares to press that he wasn't good informed of the situation. The project team was not prepared for the failure. signed up for plans using the site compared to the 227,000 that had enrolled through the 14 state run exchanges.

Obama Administration set deadline of a working site for the 'vast majority' of users. (December 1, 2013)

Failure causes & possible solutions

The failure was caused by a complex interplay of problems:

Cause 1: Contractor Management

1. Contractor delays and performance issues were not always identified.
2. A contractor incurred unauthorized costs that increased the cost of the contract.
3. Contracting officers in all government agencies did not have access to contractor past-performance evaluations when making contract awards.
4. Critical deliverables and management decisions were not properly documented.

Cause 2: Architecture, non functional requirements

1. The system was not designed to handle the massive influx of initial users.
2. Insufficient effort made to build a system that would meet the performance and availability needs of it's stakeholders.
3. Security solutions were seemingly slapped together in a shoddy manner instead of through the kind of systematic approach that is expected in a highquality software solution.
4. Many thousands of pages of legal healthcare regulations did not translated seamlessly into functional requirements.
5. The basic architecture was designed and built around the notion that the system would forward requests for quotes from insurance seekers to external vendors in real-time; however, this massive interconnectivity and the subsequent burden on the government servers caused the system to collapse.

   However, all these points, extracted from [1] are a compilation of the problem's triggers , but **where could it change?**

Besides all the disastrous organizational perspective, and scoping at the missing functional requirements, the elicitation, documentation and communication of those requirements probably could have helped this project. There are many structured techniques specifically designed to learn others about the system and what should it carry, some techniques such as 'planguage'[10] or 'volere templates'.

Cause 3: Architecture, Change Managment. The inability to handle

The inability to handle a growing amount of work, its potential to be enlarged in order to accommodate that growth, or a limited space to accommodate iterations can be a very dangerous thing. In this case, just a week before the deadline a requirement changed that had a major impact on the architectural constraints of the site.

Cause 4: Politics (Ostrich effect) - needs review. Not that great yet

Politicians were setting a deadline that was not realistic. Only two weeks for the launch there were end-to-end tests. The requirement of 60,000 users was not met according to the New York Times on 30 September 2013 that based there statements on documents released by House investigators [4]. Only 11,000 users could simultaneously access the website. Also a McKinsey & Co document, made on April 4 2013, that was adressed among others to the CMS Administrator [4, page 7]. The document stated there was insufficient time for end-to-end testing and that a 'limited initial launch' would be ideal. This report only surfaced on 13 November 2013. One and a half month after the release of the website. It is unclear why this information was not used to postpone the deadline or to do a launch for a limited number of users.

Contractor Management

Key factors in failure

One of the most important reasons for the failure of the *HealthCare.gov* project, from the process and management point of view, was a mismanagement of contractors.

The *Center for Medicare & Medicaid Services* (CMS) depended heavily on contractors to develop and manage the *Federal marketplace*. Federal law prescribes precise rules that contractor officers need to follow in order to manage the performance of the contractors. The following mistakes were made during the project [3]:

- **Contracting officers did not receive all contract deliverables and periodically neglected to use those to monitor the contractors performance.** Law enforces the contractor officers and their representatives to monitor the performance of their contractors. Sometimes reports were incomplete and deliverables handed in months after their deadlines, without any explanation requested by or given to the CMS.

- **Unauthorised CMS personnel added work and increased the cost of one of the contracts.** Contractor officers added extra work to one contract without having the authority to issue such work, around 40 work units were added for a total extra cost of $28 million.
- **Contract officer's representatives were not properly designated and authorised in written contracts.** 75% of the contracts reviewed missed important information like the contracting officer's representative or the specific duties and responsibilites assigned for each contract.
- **CMS's contracting officer representatives did not have the required certification.** Contracts that are valued at more than 10 million dollar require a Level III certifcation in risk management. Not all contracting officer's representatives, who worked with contracts of this size possesed the required certification.
- **CMS has not complied with the standards of ethical conduct in atleast one occasion.** In one case, one of the CMS panel members responsible for awarding a contract had recently worked together with the party that applied for the contract, thus creating a possible conflict of interests. The employee did not raise this as an issue to his supervisor(s) and as such acted in conflict with the CMS's ethical code.
- **Contracting officers did not always prepare contractor past-performance evaluations**. Some contracts were not registered at the instituion that keeps a record of government contractors. Subsequently, the CMS did not perform performance reviews for those contracts that are required for the logs of that institution.

Scrum

Principles applied during the project

A key part of the Scrum methodology is to prioritise communication over documentation. Due to the all the legal requirements and licensing required for big government project like this one scrum was not used as process for the overall project. Some of the contractors, like Development Seed [24], used Agile methods for their development internally. Unfortunately, the main problem for the contractors was in their communication with the CMS, who were not involved on such a low level of the project.

Which principles could help HealthCare.gov?

According to the Scrum Guide (http://www.scrumguides.org/scrum-guide.html#team) Scrum is most effective in small teams with up to 9 people. Considering the sheer size of this project, a pure Scrum approach for the entire project would have likely done more harm than good. However, some parts of Scrum could have benefitted the CMS.

Take for example a **product backlog** together with having a dedicated **product owner**; These could have stopped unauthorized CMS employees from changing the scope of the project. Any product that needs to be added to backlog would have to be added by the product owner, who can check if the request is legitimate (assuming that all product owners are authorized to change the scope of the contract).

The main problem in this case was the lack of control on the contract deliverables. Perhaps if the CMS worked with a Scrum flow which included a **demo** phase, those deliverables could be incrementally shown to the CMS stakeholders, even if they aren't software but documents e.g. security audit reports and progress reports. However, the report by Daniel Levinson [3] makes it sound like the main cause for missing these deliverables is because CMS was not enforcing them too strictly. So for this approach to yield any success, the contract officers would also need to take a more professional stance on the deliverables.

RUP

Principles applied during the project

- Vision: The vision of what to build was clear, thousands of pages of legal documents provided documentation on what to build exactly.[1]
- Plan: The product was planned to launch on 1 October 2013 and did so. Testing was planned for two weeks before launch [2]. The plan might not have been ideal, but it was there.
- Risks: Laws on contracting dictate what has to be done to supervisor contractors to mitigate risks, however those laws were not always followed [3]. This means that this principle was not followed correctly.
- Issues: Due to a lack of data, it is hard to tell what has been done in order to track issues.
- Architecture: While there was an architecture set to integrate all the services, it was not designed to handle to load that it got on the launch day.[4]
- Product: This step focuses mainly on building the product step by step and testing it during every step. In some cases this was done, with the front-end for example. The front-end was made by Development Seed and the non-interactive part of this went live before the rest of HealthCare.gov. This part of the HealthCare.gov website was generally well received and appreciated for its modern take on government websites [24]. While the system as a whole was only tested two weeks before launch. Which was

too short to find all important bugs, and too close to launch to fix all the problems that were found.

- Evaluation: This principle assumes an interative approach, which was not present during the building of HealthCare.gov. So this principle was not followed at all.
- Change Requests: The Obama administration kepy modifying regulations and policies of the Affordable Care Act, which inherently meant that the scope of the HealthCare.gov website kepy changing too. Which probably has caused some hardship with the changes trickling down to all the contractors, but there is a lack of documentation on how the changes were documented and implemented.

Which principles could help HealthCare.gov?

As stated above, a stricter monitoring of the deliverables by the contractors could have helped this project. What also could have prevented the failure was a better focus on product and evaluation. If a more Agile way of working was promoted, with small deliverables, that can be tested. Then perhaps bugs and the wrong scale for the architecture could have been identified earlier, while there was still time to fix the problems. Instead of when the integration happened, two weeks before the launch.

Mars Lander

The Mars Lander's development approach focuses on durability and reliability to the extreme. Judging by how the HealthCare.gov website only worked correctly for 1% of the visitors during their launch week, this was not the case for the HealthCare.gov website.

Every part that is aboard the Mars Lander needs to go through a qualification testing procedure of 12 to 15 months. And then when all the components are combined there is another qualification testing procedure which can take a few years [5]. This shows that the Mars Lander team takes their integration tests very seriously, which was not the case for the CMS with all the missing deliverables.

Ofcourse there is a downside to this as well. If the CMS was to implement qualification testing on this scale, costs for the project would likely skyrocket and the project would take longer. Then again, the HealthCare.gov is not flying through space, where maintenance is impossible to perform. So the CMS could definitely profit by looking how NASA does its integration tests and implementing those, albeit on a smaller scale.

Architecture - rigid and lacking non-functional requirements

Key factors in failure

One of the primary problems with HealthCare.gov was its performance. The architecture was designed to pass on requests from the web servers real-time to the back-end systems. The system was not designed to be able to handle the high loads that it faced after going to production [13].

Requirements that had a high impact on the architecture were delivered late. One such requirement, was that users needed to log in, before they could use the site: “[customer representatives were] debating whether consumers should be required to register and create password-protected accounts before they could shop for health plans”. This was delivered three months before the planned release date. There was scant time, to incorporate these requirements [9].

Another one of the problems the site faced, was that there were a number of security vulnerabilities [18]. These came to light during a security breach in the July of 2014. The breach was discovered by a CMS security team in the following month. It occurred due to a development server not being properly configured and this led to malware being uploaded to the system.

Software development methodologies

Could the problems stated above have been fixed by using software methodologies?

Scrum

The back-end of HealthCare.gov is reported to have been built using agile practices but they were likely not applied correctly due to the project having explicit phases defined such as testing [12]. In Scrum, architecture is designed as required for the next production increment [16]. This does not necessarily lead to the architecture becoming more flexible. If a Scrum approach would have been implemented properly, a minimal viable product would have been created first, which should have already contained the performance requirements. The different teams working on the project should have agreed on a definition of done. A large amount of work on HealthCare.gov was left unfinished during the launch [20]. A clear and formal definition of done would have resulted in an architecture containing all quality attributes of the system. However, a minimal viable product might not have included integration with all systems, which is where the performance really starts to hurt.

On the other side of the spectrum, the front-end of HealthCare.gov is considered to be a success. A start-up known as Development Seed was responsible for the front-end of the website and made use of agile practices [12].

Rational Unified Process

Contractors stated "…mere days before the launch date, [CMS] tested the system's ability to handle tens of thousands of users at once. It crashed after a few hundred." [9]. This likely indicates an issue with validating the architecture.

Next to iterative development, RUP also defines phases of a project's lifecycle [15]. In the elaboration phase, most of the architecture should be done. At HealthCare.gov most of the architecture was also done up front. This did not help them forsee the problems that occured during the go-live. RUP does promote a risk first approach. If RUP would have been properly implemented, the riskier parts of the process, such as integration and performance, would have been tackled in the beginning.

MARS Rover

The software methodology used to build the Mars rover is based on risk-reduction principles [19]. The U.S. government has built a large number of software systems and thus has a history of security vulnerabilities that can possibly occur. One of the security vulnerabilities found was due to a development system being configured with default credentials which is a mistake made by the developers. This made it possible for an automated attacker to take advantage of the system and thus use it to upload malware [18]. Apparently this weak-link in the security was not identified in the architecture of HealthCare.gov. If CMS were to have taken measures to reduce the amount risk involved when building the system, this would have have likely prevented the vulnerability, simply due to it being an extremely common error. The development system would have been properly identified as a point of weakness. Because of this, it would not have been insecurely configured.

Change Management

In order to shop for a healthcare plan, consumers had to register on the site for an account first. It was reported that the response times of the registration pages were very high with load times up to 71 seconds [9,21]. This symptom occured because architecture was not able to handle the concurrent users for the registration pages. Around September 2013, just weeks before the deadline, a decision was made that all users had to register first, before they could start shopping for a health plan [22]. Because of this change the registration pages became a bottleneck for the entire site. The changing functional requirement had a major impact on the architecture of the site.

Key factor in failure

Many failures regarding performance were due to a poor architectural design in the first place which in turn was caused by the omission of well defined quality constraints [1]. This was discussed in the previous chapter. However, the cause for the performance problems at the registration pages was a requirement that changed in a late stage during the development. The requirement had a major impact on the architectural constraints of the site that was not captured. Changing requirements during the course of a project is a common phenomenon in software engineering. We analysed whether Scrum or RUP have proper mechanisms to deal with this and if these mechanisms could have prevented the problems at HealthCare.gov.

	Description
Symptom	High response times for the registration pages.
Failure	The architecture could not handle the many concurrent users.
Cause	Three months before the deadline a requirement changed that stated that consumers must register before able to shop for health plans.
Context	Changing requirements with major architectural impact.
Problem	No mechanism to deal with changing requirements.

Scrum

In Scrum requirements are expressed in Product Backlog Items and are maintained by a Product Owner who is also the sole person reposonsible for them. These Product Backlog Items can change any time up until the moment they become ‘in sprint’ and are being developed by the Development Team. It is during the Sprint Planning at the start of each Sprint that the Development Team descides on how to implement the Product Backlog Items requested by the Product Owner [17]. Scrum however does not specify what the ‘how’ should answer in terms of quality constraints. Capturing the impact of the change at HealthCare.gov would have been greatly dependent of the expertise of the Development Team or Product Owner.

Another principle of Scrum is that at the end of each Sprint the team delivers a fully tested and deployable product. In general, if this were the case at HealthCare.gov the performance failure would still have occurred but it would be detected within the duration of a sprint. The project management would than have more time to handle the problem. Note however, that this specific requirement change was made weeks before the final deadline so having sprints would have made no difference either.

Overall Scrum does not provide well defined mechanisms to handle quality assurance needed when dealing with changing requirements.

RUP

The Rational Unified Process (RUP) is an iterative software development process in which risk management is an important feature. One of the ten essentials 'Change Requests' specifically deals with changing requirements by providing a process for risk and impact analysis for any proposed change [23].

> "The benefit of Change Requests is that they provide a record of decisions, and, due to their assessment process, ensure that impacts of the potential change are understood by all project team members. The Change Requests are essential for managing the scope of the project, as well as **assessing the impact of proposed changes.**" [23]

It is this part of RUP that could have prevented the performance problems during registrations. Had the change been properly analysed on its impact it could have on the architecture, the project team could have decided to make the proper adjustments on the architecture.

Sources

[1] Cleland-Huang, Jane. Don't Fire the Architect! Where Were the Requirements? Software, IEEE 31.2 (2014): 27-29.

[2] Morgan, David & Humer, Caroline. "Timeline: U.S. healthcare law's technology breakdown" Reuters, 30 October 2013, Web. 9 February 2015.

[3] Levinson, Daniel. "CMS Did Not Always Manage and Oversee Contractor Peformance for the Federal Marketplace as Required by Federal Requirements and Contract Terms", Office of Inspector General, September 2015, Report.

[4] Anthopoulos, L., et al.Why e-government projects fail? An analysis of the Healthcare.gov website, Government Information Quarterly (2015).

[5] Harwood, William. "Slow, but rugged, Curiosity's computer was built for Mars" CNET, 10 August, 2012, Web. 12 February 2015.

[6] [The Office of Audit Services (OAS) provides auditing services for HHS](http://oig.hhs.gov/oas/reports/region3/31403001.pdf).

[7] [The Failure of HealthCare.gov Exposes Silicon Valley Secrets](http://www.computer.org/csdl/mags/ic/2014/

[8] Government Accountability Office (GAO) (2013). "Patient protection and affordable care act: Status of CMS efforts to establish federally facilitated health insurance exchanges." (GAO-13-601), 19 June 2013.

[9] Chambers & Associates Pty Ltd, Case Study Analysis Saving Obamacare.

[10] Tom Gilb (Planguage). (2001) Intel Corporation by Erik Simmons "Quantifying Quality Requirements Using Planguage".

[11] [Full Committee Hearing - Is My Data on Healthcare.gov Secure?](https://science.house.gov/legislation/hearin
committee-hearing-my-data-healthcaregov-secure), transcribed here.

[12] A. Jeffries, "Why Obama's Healthcare.gov launch was doomed to fail", The Verge, 2013.

[13] USA TODAY, "Obama adviser: Demand overwhelmed HealthCare.gov", 2016.

[14] M. Heusser, "6 Software Development Lessons From Healthcare.gov's Failed Launch", CIO, 2016.

[15] P. Kruchten, The rational unified process. Reading, Mass: Addison-Wesley, 1999.

[16] Schwaber, Ken. Scrum development process, 1997.

[17] Ken Schwaber, Jeff Sutherland, "The Scrum Guide", Scrum Alliance, 2013.

[18] Cbsnews.com, "Critical" flaw found in HealthCare.gov security, 2016.

[19] A. Scoica, "Profile Benjamin Cichy", XRDS: Crossroads, The ACM Magazine for Students, vol. 20, no. 3, pp. 68-69, 2014.

[20] B. Ehley, "After 2 Years and $2.1 Billion, HealthCare.gov Is Unfinished", The Fiscal Times, 2016.

[21] AppDynamics, Jim Hirschauer, "Technical deep dive on what's impacting Healthcare.gov", 2013.

[22] The New York Times, "From the Start, Signs of Trouble at Health Portal", 2013.

[23] Leslee Probasco, "The Ten Essentials of RUP - the Essence of an Effective Development Process", Rational Software White Paper, 2002.

[24] Ford, Paul. "The Obamacare Website Didn't Have to Fail. How to Do Better Next Time", 16 October 2013, Web. 13 February 2015.

[25] Robertson. "Mastering the Requirements Process" Pearson Education (Us) augustus 2012, Chapter 16, page 353.