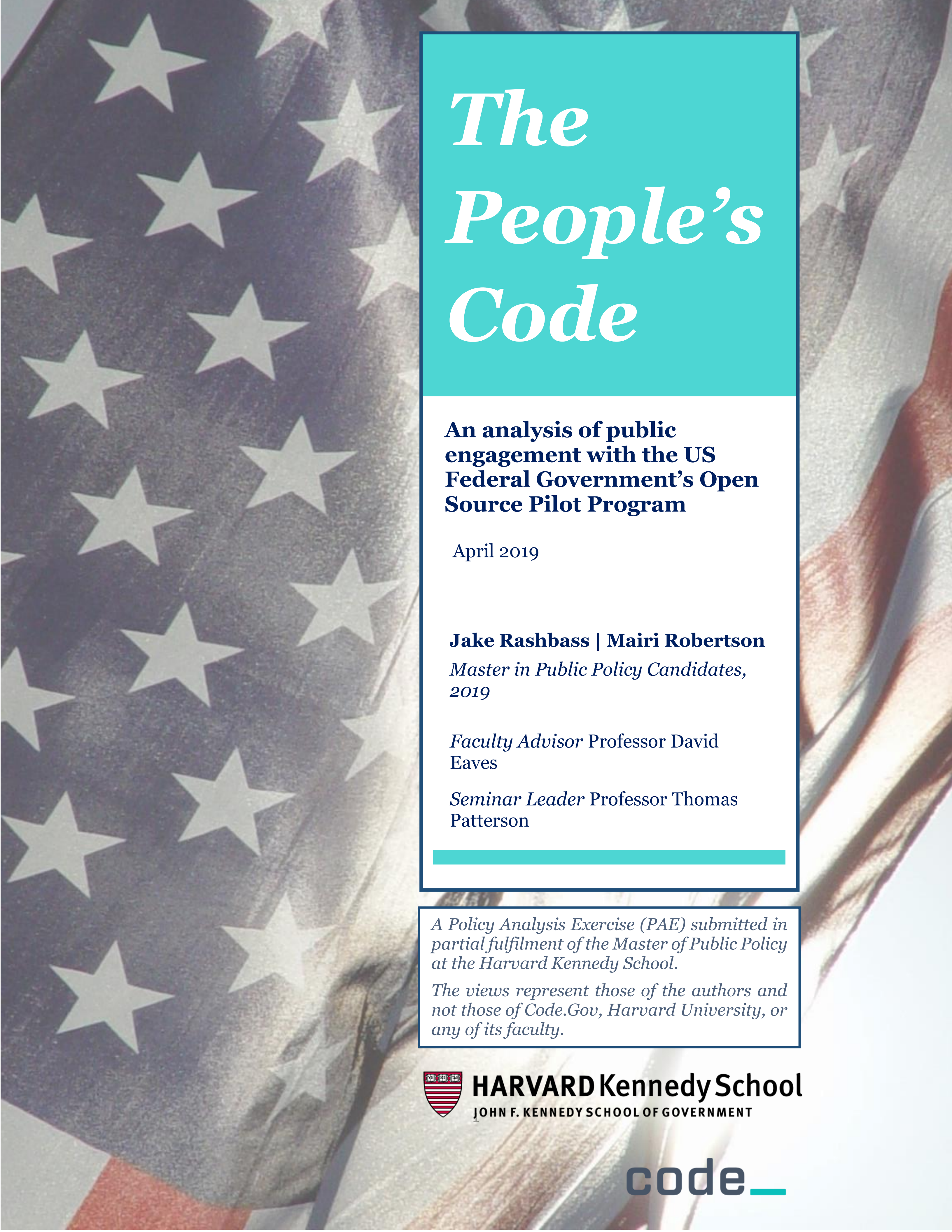


The People's Code

An analysis of public engagement with the US Federal Government's Open Source Pilot Program

April 2019

Jake Rashbass | Mairi Robertson

Master in Public Policy Candidates, 2019

Faculty Advisor Professor David Eaves

Seminar Leader Professor Thomas Patterson

HARVARD Kennedy School
JOHN F. KENNEDY SCHOOL OF GOVERNMENT

code_

GLOSSARY

DOD	Department of Defense
GSA	General Services Administration
LISH	Laboratory for Innovation Science at Harvard
OMB	Office of Management and Budget
OSI	The Open Source Initiative
OSS	Open Source Software
PAE	Policy Analysis Exercise
RfP	Request for Proposal
SLOCs	Software Lines of Code

EXECUTIVE SUMMARY

THE PROBLEM

In August 2016, the Obama Administration launched the Federal Source Code Policy. It aimed to promote the use of open source software in the federal government by requiring that federal agencies:

1. ***Share all custom source code with one another***, to reduce duplicative software procurement spending.

2. ***Share 20% of their custom source code with the public in an 'Open Source Pilot Program,'*** to allow developers outside government to contribute to government software projects and to allow taxpayers to reuse the code that they pay for.

Code.Gov – our client – was created by the White House to lead this policy. Part of their brief was to figure out how to stimulate public engagement with federal source code released through the Open Source Pilot Program.

The Pilot Program will expire in August 2019. Before that deadline, Code.Gov must decide what changes – if any – they should make in the next iteration of the policy to maximize public engagement with the government's open source code.

RESEARCH QUESTIONS

1. ***To what extent did the Pilot Program increase user engagement with federal source code – and how did users engage?***
2. ***Which factors drove user engagement with federal source code?***
3. ***What changes should Code.Gov make – if any – to further boost user engagement in the second iteration of the Program?***

METHODOLOGY

'Big data' quantitative analysis of almost 200,000 engagements with over 5,000 different federal open source projects since 2008

+

Qualitative analysis of 10 expert interviews, 2 focus groups involving 12 federal employees, and literature review

FINDINGS AND RECOMMENDATIONS

1

At the aggregate level, the Pilot Program increased neither the rate at which federal open source projects were created nor the rate at which users engaged with those projects. However, a small number of projects were highly successful: The top 0.04% of projects accounted for over 40% of engagement. They are followed by a long tail of projects which had little-to-no user engagement.

2

Nine characteristics drive user engagement with public source code, including a project's discoverability, reuse potential, and documentation. We combine these into the 'DREAM CODE' framework.

3

If Code.Gov decides to turn the Pilot into a permanent program after the August 2019 expiry date, it should make four changes to the program assuming that it seeks to maximize user engagement:

a. Articulate the purpose of the policy clearly. Today, there is confusion amongst agencies and industry about whether user engagement is an objective of the program. This impacts how it has been implemented.

b. Adopt a 'default to open' policy requirement. Today, agencies are only required to release 20% of their code to the public. This is difficult to enforce and means agencies don't always release code that would most drive public engagement.

c. Introduce additional programmatic support to agencies. Many agencies do not have the institutional expertise or capacity to meet the requirements of the Federal Source Code Policy. This impacts their ability to drive user engagement. Code.Gov should ensure they have additional support – for example, mandating that agencies hire community engagement managers – to rectify this.

d. Investigate outstanding questions raised by this report. In our research, we have uncovered important questions – for example, "Who are the users that engage with federal source code?" - that we were unable to answer due to citizenship and technical constraints. Code.Gov should prioritize finding answers to these questions as it considers designing the next stage of the Federal Source Code Policy.

CONTENTS

1. INTRODUCTION

The US federal government is the world's largest single purchaser of code,[1] spending over $6 billion on software every year.[2] This software is critical for the federal government: It forms the backbone of everything from rocket launches to disease control programs.

The challenge for policymakers is how to provide this technological capacity while managing costs. One part of the solution is Open Source Software (OSS), which powers 90% of the world's computer applications according to estimates.[3] In this section, we provide an overview of OSS and its role in federal government software purchases. We also explain why the Obama Administration decided to promote OSS through 2016's Federal Source Code Policy – and how this report seeks to inform the design of the Policy's future.

[1] Kesan, Jay P. & Shas, Rajiv C. Shah, Shaping Code, *Harvard Journal of Law & Technology*, Volume 18, Number 2 Spring 2005 p 373.

[2] 'Improving the Acquisition and Management of Common Information Technology: Software Licensing'. Office of Mgmt. & Budget, Exec. Office of the President, June 2, 2016. p. 1. Available at: https://obamawhitehouse.archives.gov/sites/default/files/omb/memoranda/2016/m-16-12_1.pdf.

[3] Zorz, Zeljka. "The Percentage of Open Source Code in Proprietary Apps Is Rising." *Help Net Security*, 22 May 2018, www.helpnetsecurity.com/2018/05/22/open-source-code-security-risk/.

FEDERAL SOFTWARE PROCUREMENT

The federal government's software purchases can take one of three forms:

- ***Proprietary***, or 'closed source software,' meaning the organization from which the code was procured retains the intellectual property rights for the code;
- ***Open source***, meaning the copyright holder grants anyone the right to use, alter, and distribute the code for any purpose; or
- ***Mixed source***, in which some of the code – or accompanying support services – is provided on a proprietary basis while the remainder is open source.

An additional dimension – and which is not mutually exclusive from the categories above – is ***custom code***. This refers to situations in which the federal government orders bespoke solutions to be developed for its technical requirements. These are usually either proprietary or mixed source solutions, meaning they incur large costs for the taxpayer. However, they can also be developed as OSS.

THE CASE FOR OPEN SOURCE

Proponents of open source cite five major reasons for using OSS in the federal government.[4]

First, proprietary software is usually more expensive. Licenses for proprietary software are generally priced on an annual basis and are contracted over multiple years. This means that organizations are locked-in to significant software expenditures over long periods. Organizations that use proprietary solutions are often also required to pay for on-going maintenance and support. Support is often charged regardless of use, meaning that even if the purchaser does not need significant support, they pay for it. This can cost ~20% of the initial software price and is charged annually. [5]

Together, these recurring license and maintenance costs add up to the $6bn figure cited earlier. Individual software and support contracts can be significant outlays for federal agencies. For example, the US Department of Defense (DOD), Coast Guard, and intelligence community pay ~$350m per year to Microsoft Enterprise Services for licensing and support – just one of those agencies' many IT procurement contracts.[6]

OSS solutions are often – although not always – cheaper. If they already exist and don't need to be built from scratch, they are available for free, meaning there is no initial outlay. Moreover, they do not require licenses for ongoing use. While there may be maintenance costs – for example, hiring programmers to develop additional capabilities in the software – these are paid on an as-needed basis.

The US federal government has more than 42,000 different software licensing and support contracts. [7] This almost certainly means there is excessive spending on licenses for software that could be replaced with OSS. There is also excess spending on 24/7 support that is not fully utilized. While

[4] These are compiled from our series of 10 interviews, two focus groups, and extensive literature review.

[5] "Proprietary Software vs. Open Source - The Hidden Costs." *Trellon*, trellon.com/content/blog/proprietary-software-vs-open-source-hidden-costs.

[6] U.S. Department of Defense. "Contracts for January 11, 2019." *U.S. DEPARTMENT OF DEFENSE*, dod.defense.gov/News/Contracts/Contract-View/Article/1730557//.

[7]Kuldell, Heather. "It's Official: MEGABYTE Act Signed into Law." *Nextgov.com*, Nextgov, 28 Nov. 2017, www.nextgov.com/cio-briefing/2016/08/its-official-megabyte-act-signed-law/130391/.

major IT infrastructure such as Office 365 will likely continue to be bought on a proprietary basis, OSS solutions could replace a non-trivial proportion of those 42,000 transactions.

Second, OSS prevents vendor lock-in. When organizations purchase proprietary software, contract terms typically force the organization to rely on the original developer for modifications, support, and updates. This puts the developer in a powerful position: They can charge significant sums for bespoke service. This is especially a problem in the public sector, where generic solutions are often tailored to suit the government's requirements around scale, security, and specificity. It also gives the vendor leverage in negotiations for future contracts: migrating from one software platform to another is difficult, particularly in government where service outages can have significant consequences. The result is that government IT procurement managers may be reluctant to seek alternative solutions.

OSS eliminates much of this problem. Because the code is public, any developer – whether the original developer, government employee, or an external consultant – can make changes. This allows government to hire or contract developers who are cost-efficient and can do the task at hand. OSS also provides an insurance policy against the extreme case in which the original developer goes out of business.

Third, OSS improves the reliability and security of software solutions. The open source method is a form of 'peer review,' in which developers from different industries and countries can improve software and check it for bugs. Having more eyes on code generally improves its quality: Studies find that OSS has fewer bugs, on average, than proprietary software.[8] The additional review and vetting also means it is easier to verify the security of OSS, since proprietary software must be vetted by the developer – leaving the end-user reliant on a single actor to test security concerns.

Fourth, OSS can be easily shared. A federal agency that develops an OSS solution can publish the code online, meaning that other agencies – not to mention state and local governments, non-profits, and civil society – can also use the solution. The states of California and New York, for example, have used the Data.Gov source code as part of their own statewide open-data resources.[9] Similarly, the source code behind a federal government analytics website which presents a public dashboard of web traffic on government websites, Analytics.Usa.Gov, has been reused by several state and local governments across the United States since being open sourced.[10]

This helps reduce the duplication of federal software procurement. Experts estimated in 2016 that almost one-third of the federal government's $6bn in software spending was duplicative or wasteful.[11] One way to avoid this situation is to catalogue software

[8] From the most recent annual Coverity Scan, which looks at approximately 1 billion lines of open source and proprietary code. Summary available at Rubens, Paul. "Open Source Code Contains Fewer Defects, But There's a Catch." *CIO*, CIO, 18 Nov. 2014, www.cio.com/article/2847880/open-source-code-contains-fewer-defects-but-theres-a-catch.html.

[9] See "Open Data | Open Data NY." *State of New York*, data.ny.gov/ and "Data.ca.gov." *Data.ca.gov*, data.ca.gov/.

[10] Mill, Eric, et al. "Digital Service Delivery | How We Built Analytics.usa.gov." *18F*, 19 Mar. 2015, 18f.gsa.gov/2015/03/19/how-we-built-analytics-usa-gov/.

[11] Prior to the introduction of the MEGABYTE Act in late 2016. Some of this duplicative spending may since have been reduced – it is too early to tell how successful the policy has been. Please see Goldstein, Phil, "Federal Agencies Will Be Required to More Accurately Track Software Licenses." *Technology Solutions That Drive Government*, 24 Aug. 2016, fedtechmagazine.com/article/2016/08/federal-agencies-will-be-required-more-accurately-track-software-licenses.

licensing agreements across government, so that agencies can avoid procuring software that has already been purchased elsewhere.[12] Another is to encourage the use of OSS, so that even if a federal agency fails to recognize that another agency has already procured their desired solution, they are not spending additional money on it.

Fifth, OSS is fast. Projects developed in the open from the outset benefit from the myriad of potential contributors in the open source community. Patches and bug fixes can be created by anyone, and don't require the organization to wait for the original vendor to work on it. This means that the development cycle is generally shorter and more agile compared to proprietary solutions.

This benefit is particularly important in the public sector. The rapid prototyping that comes with agile software development is easier in an OSS context, meaning that agencies can test and adjust software prior to deploying it in government services. If there are bugs after an agency has adopted an OSS solution, the patching process is more rapid – reducing the interruption to government services.

REASONS FOR CAUTION

It is important to note that OSS is not a silver bullet for all government software procurement.

There are cases where proprietary solutions make more sense for federal agencies, and even advocates of a 'default to open' policy[13] usually accept that there are cases where exceptions should be made. For example, this includes cases where:

- Sharing source code could create national security or individual privacy risks. This could apply to some military applications, where the federal government may have a legitimate interest in not sharing software with other actors;
- Sharing source code might damage an agency's systems, personnel or programs;
- The government needs guaranteed and on-call support; or
- The law restricts source code being shared, for example under Export Asset Regulations or International Traffic in Arms Regulation.[14]

This report does not argue that OSS should be used in all circumstances. Instead, the underlying assumption is that OSS is underutilized by federal agencies given the benefits of this approach.

THE (ORIGINAL) PROBLEM

Until August 2016, the procurement landscape described above presented two challenges for the Federal Government.

First, purchases of custom code by one agency were in many cases duplicative. Agencies would order bespoke software when code procured by other agencies could have filled the same requirements. Without any impetus to share custom code purchases between agencies, the federal government was double-spending on technical requirements. This situation led to the ~$2bn in excess spending cited earlier in this chapter.

Second, purchases made by the federal government are – on some level – the property of the taxpayers

[12] The MEGABYTE Act of 2016 attempted to do this.

[13] i.e. all government software procurement should be open source.

[14] These examples are specifically cited in OMB's Federal Source Code Policy: "Federal Source Code Policy." *Exceptions to Government Code Reuse*, sourcecode.cio.gov/Exceptions/.

who fund those purchases. However, taxpayers were not reaping the benefits of that code. In the words of the Obama Administration, the procurement process was wrongfully hiding "the people's code."[15] This meant the private sector and civil society could not utilize publicly-funded code for commercially- and economically-productive purposes.

THE POLICY RESPONSE

In August 2016, the Obama Administration announced the 'Federal Source Code Policy.'

It created a new team within the Office of Management and Budget (OMB) in the Executive Office of the President, Code.Gov, to spearhead this policy. Code.Gov is the client for this PAE. This team was transferred over the course of our research to the General Services Administration (GSA).

The Federal Source Code Policy contains two pillars that address the problems laid out above. These are:

1. ***Interagency requirement***: Any new custom code developed by or for the federal government must be made available for all federal agencies to use; and
2. ***Public requirement***: At least 20% of custom code must be released as open source software to the general public (Appendix A).

The public requirement contains complexities that are not immediately obvious. For example, the policy does not stipulate how to measure '20%' of code: 'Volumes' of source code are not easily quantifiable, and the policy states that agencies can devise their own metrics to determine whether they have shared 20% of code. Potential metrics could include the total number of projects, files, Software Lines of Code (SLOCs), compiled data size, contract actions, or dollar value. Opting for one metric over another could produce very different results.

The 20% requirement also means that agencies are incentivized the release the code that is 'easiest' to open source, rather than that which maximizes public benefit. Although the policy states that agencies should release code which it "considers potentially useful to the broader community," there is little guidance around what this means in practice.[16] Our empirical analysis in Chapter 3 – which finds that there has been little public engagement with federal source code released under the policy – suggests that this vagueness has affected the success of the policy.

Given these complexities – and the federal government's relative lack of experience in OSS – Code.Gov decided to pilot the second pillar of the Federal Source Code Policy in order to test whether it could succeed and to understand how to improve its performance. The Open Source Pilot Program ('the Pilot Program') was launched in August 2016, to last for three years.

Today, the end of the Pilot Program – August 2019 – is rapidly approaching. Before the pilot ends, Code.Gov must decide whether to turn it into permanent policy – and, if so, what alterations should be made.

[15] "The People's Code." *National Archives and Records Administration*, National Archives and Records Administration, obamawhitehouse.archives.gov/blog/2016/08/08/peoples-code.

[16] "Federal Source Code Policy." *Open Source Software*, sourcecode.cio.gov/OSS/.

THE ROLE OF THIS REPORT

Our research provides empirical data and qualitative analysis to aid Code.Gov in its decision about whether to continue the Pilot Program – and, if so, how it should do so.

We analyze how the public has engaged with the code which has been open sourced by federal agencies.[17] Because the Pilot Program was established to give the public access to federal government code, understanding whether the public has indeed engaged with code released in the Pilot is vital for measuring the Pilot's success.

We also investigate the reasons driving the success of the OSS projects that have received the most engagement. We find nine characteristics that are linked with successful OSS projects.

To this end, this PAE has two major analytical components:

1. ***A quantitative analysis of the public's engagement with open source code released by the federal government.*** We investigate the extent to which the Pilot Program led to an overall increase – or otherwise – in engagement with federal OSS projects. We also analyze which agency sub-departments and projects received the most engagement.
2. ***A qualitative analysis of the repositories that receive the most engagement***. We draw on expert interviews, focus groups with federal employees, and a literature review to understand why certain repositories and agencies have outperformed others.

Based on these analyses, we provide a set of recommendations about how to improve public engagement with source code released by the federal government. This includes codifying the best practices used by the top-performing federal repositories so that they can be applied to other federal repositories.

[17] We do *not* address the first pillar of the Federal Source Code Policy, which covers interagency code sharing. This is because as government outsiders, we can only access public information and do not have the capacity to communicate with every federal agency to provide a rigorous and useful analysis of agencies' experience of the Pilot Program.

2. METHODOLOGY

We have developed an original and substantial dataset for this analysis. It includes data for almost 200,000 interactions with >5,000 different OSS projects owned by the 23 federal agencies that the Federal Source Code Policy applies to.[18] This is the first time such a dataset has been developed and made available to GSA.

To understand how the public has engaged with source code shared by the federal government, we turned to GitHub, an online platform where the vast majority of federal agencies' open source code is located.[19] We created custom code to scrape GitHub for information about federally-owned projects.[20] In this section, we explain in more detail how the dataset was developed and what information it contains. We also outline some of the limitations it has. Finally, we explain how the dataset is combined with our qualitative analysis to produce the recommendations at the end of this report.

[18] Those covered by the Chief Financial Officers Act of 1990.

[19] From multiple interviews including Code.Gov team management and staff.

[20] A copy of this code is provided in Appendix B. It was developed with the support of Joseph Castle (of Code.Gov) and Froilan Irizarry (formerly of Code.Gov, and now of GitHub).

OVERVIEW

This report analyzes the success of the Pilot Program to date and provides recommendations about how to improve it going forward.

To do this, we combine a 'big data' quantitative analysis with qualitative research. In the former, we develop an original dataset to measure the amount of engagement with federal OSS projects to date. This is the focus of Chapter 3. We find that at the aggregate level, the rate of engagement with federal open source has not increased since the policy came into place.

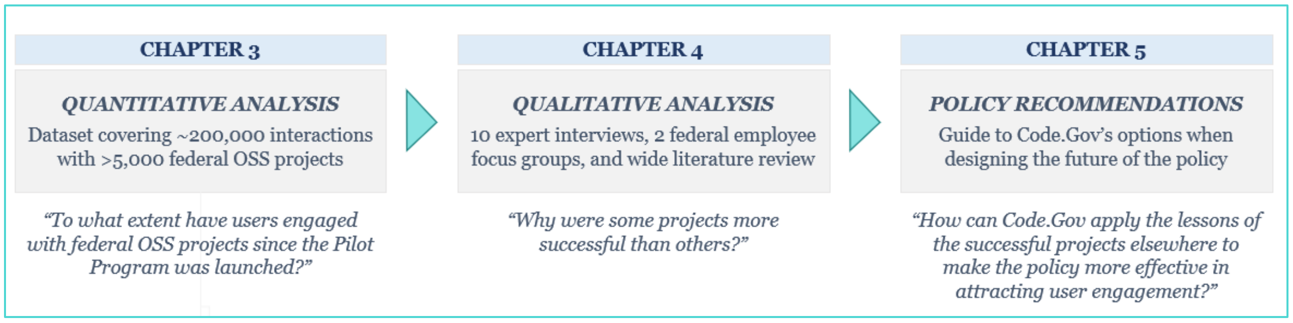
However, some projects have performed well. We seek to understand why these projects out-performed their peers in order to identify 'best practices' that could be adopted in other federal OSS projects. This is the subject of Chapter 4. The theory underlying this approach is that adopting these best practices would help increase aggregate engagement with federal OSS projects.

To identify these best practices, we conducted a wide-ranging review of qualitative sources including expert interviews, focus groups with federal employees, and a literature review. From these sources, we distilled a list of the characteristics which drive successful government OSS projects.

Finally, we produce a series of recommendations about how the Pilot Program should be changed after August 2019 so that public engagement with federal OSS projects increases. These are included in Chapter 5. The recommendations are based on what we learnt from our conversations during the qualitative component of our research.

An illustration of the methodology underpinning this report is provided below in Figure 1.

Figure 1: Overview of methodology


CHAPTER 3
QUANTITATIVE ANALYSIS
Dataset covering ~200,000 interactions with >5,000 federal OSS projects
"To what extent have users engaged with federal OSS projects since the Pilot Program was launched?"
CHAPTER 4
QUALITATIVE ANALYSIS
10 expert interviews, 2 federal employee focus groups, and wide literature review
"Why were some projects more successful than others?"
CHAPTER 5
POLICY RECOMMENDATIONS
Guide to Code.Gov's options when designing the future of the policy
"How can Code.Gov apply the lessons of the successful projects elsewhere to make the policy more effective in attracting user engagement?"

QUANTITATIVE ANALYSIS

GITHUB AND USER ENGAGEMENT

GitHub is the leading platform for software developers to share code and to collaborate on it.

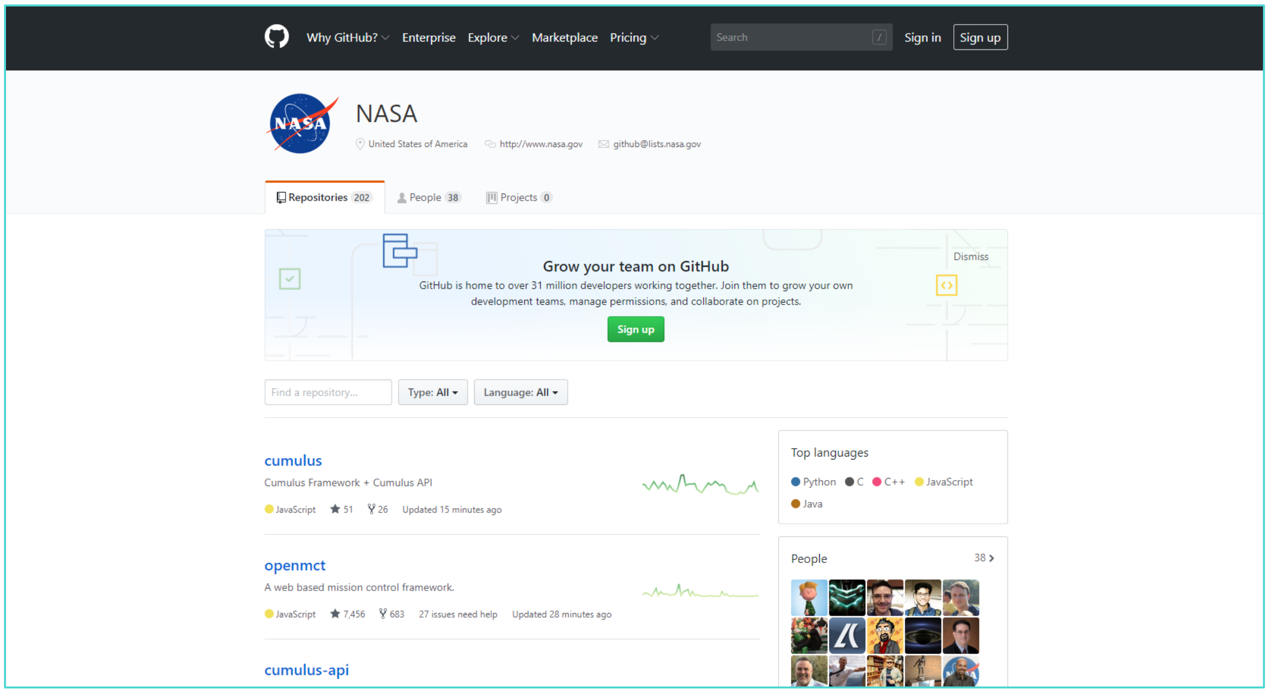
Every federal agency has a GitHub account – and in some cases multiple accounts – where they share source code projects they have developed or commissioned. Each project is called a "repository" (Figure 2). In this way, other GitHub users, who range from software developer professionals to amateurs and students, can discover and interact with federal code for their own use. Figure 3 shows an example result of what the code in a repository looks like when developed into an application.

There are multiple ways through which users can engage with another user's or organization's repositories. For the purposes of this analysis, we have captured data on four forms of engagement. These are:

- ***Stars*** – A 'star' is created when a user marks a repository they find interesting. Starring a repository allows the user to keep track of its developments, discover similar content on GitHub, and show appreciation for the content to the repository owner. In the context of our research, stars can be interpreted as users marking repositories shared by the federal government that they find particularly interesting and/or useful.
- ***Forks*** – A 'fork' is created when a user makes a personal copy of another user's repository for their own account. In the context of our research, this means that a user is directly copying a federal repository for their own use.
- ***Issues*** – An 'issue' is created when a user provides feedback on a repository. This feedback could mean reporting a software bug, suggesting an improvement, or asking a question. Repository maintainers and other users can then respond to the issue. An issue is marked as 'closed' once it has been resolved. In the context of our research, issues are a way for repository owners to gain user feedback on the repository.
- ***Pull requests*** – A 'pull request' is created when a user submits a proposed change to a repository. This differs from 'issues', in that the user actually submits a proposed alternative version of the code (rather than simply alerting the owner to a potential flaw in the code). With pull requests, the repository owner can then either accept or reject the proposed change. In the context of our research, pull requests are a means for repository owners to directly incorporate contributions from other GitHub users.

In this PAE, we define 'public engagement' as the total number of stars, forks, issues, and pull requests that GitHub users (the "open source community") have generated in their interactions with open source code shared by the federal government on the GitHub platform.

Figure 2: The home page for NASA's library of repositories


Why GitHub? Enterprise Explore Marketplace Pricing Search Sign in Sign up

NASA

United States of America http://www.nasa.gov github@lists.nasa.gov

Repositories 202 People 38 Projects 0

Grow your team on GitHub

GitHub is home to over 31 million developers working together. Join them to grow your own development teams, manage permissions, and collaborate on projects.

Sign up Dismiss

Find a repository... Type: All Language: All

cumulus

Cumulus Framework + Cumulus API

JavaScript ★ 51 26 Updated 15 minutes ago

openmct

A web based mission control framework.

JavaScript ★ 7,456 683 27 issues need help Updated 28 minutes ago

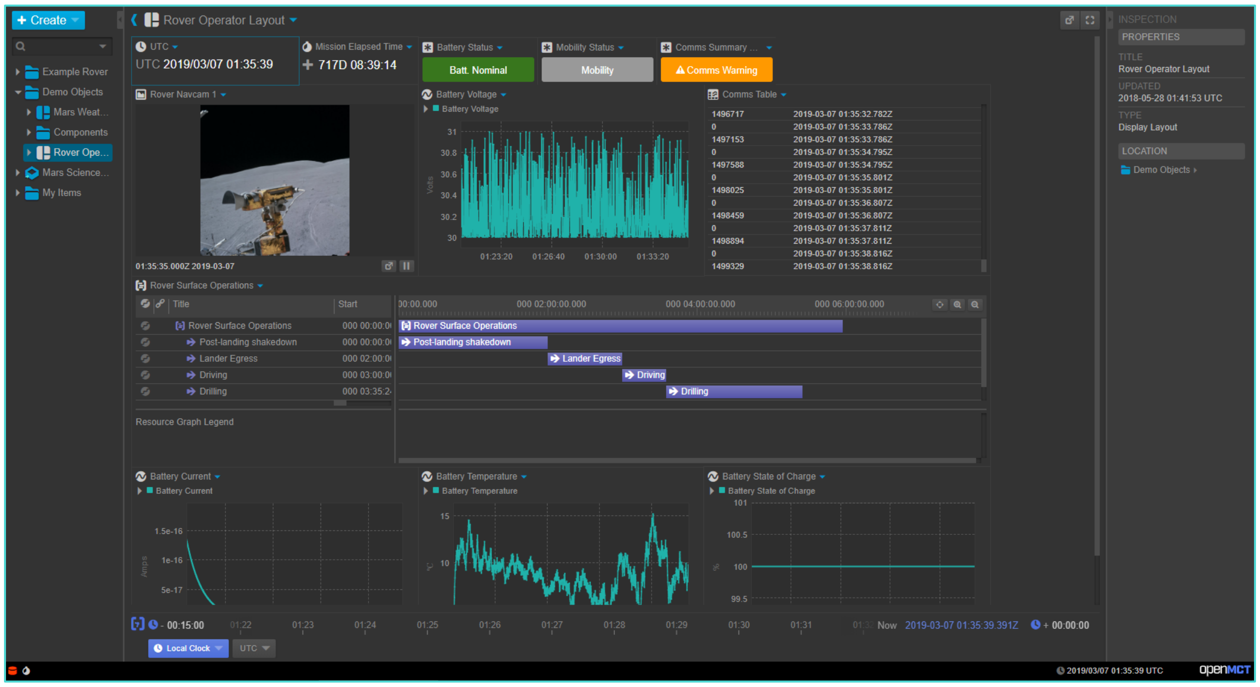
cumulus-api

Top languages

● Python ● C ● C++ ● JavaScript ● Java

People 38

Figure 3: One of NASA's open source applications



Comms Table

1496717	2019-03-07 01:35:32.782Z
0	2019-03-07 01:35:33.786Z
1497153	2019-03-07 01:35:33.786Z
0	2019-03-07 01:35:34.795Z
1497588	2019-03-07 01:35:34.795Z
0	2019-03-07 01:35:35.801Z
1498025	2019-03-07 01:35:35.801Z
0	2019-03-07 01:35:36.807Z
1498459	2019-03-07 01:35:36.807Z
0	2019-03-07 01:35:37.811Z
1498894	2019-03-07 01:35:37.811Z
0	2019-03-07 01:35:38.816Z
1499329	2019-03-07 01:35:38.816Z

Rover Surface Operations

Title	Start
Rover Surface Operations	000 00:00:0
Post-landing shakedown	000 00:00:0
Lander Egress	000 02:00:0
Driving	000 03:00:0
Drilling	000 03:35:2

OUR DATABASE

Our database contains information on every engagement that GitHub users have made with each repository in the database, along with the repository's owner agency, and creation date.

A summary of the database is included in Table 1.

This dataset represents the first significant 'big data' analysis of the Federal Source Code Policy. McKinsey & Company defines 'big data' as a dataset "whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze."[21] The complicated process required to generate this dataset – which included writing original script that 'scraped' data from GitHub – and the almost 200,000 datapoints it contains means that it falls within this definition.

This is significant. Until now, Code.Gov has only had access to 'supply side' information about the policy – that is, data that summarizes the percentage of federal agencies complying with the Federal Source Code Policy.[22] This dataset improves the ability of the Code.Gov team to understand the 'demand side' – specifically, the trends and drivers of public engagement with federal source code.

LIMITATIONS OF APPROACH

Our approach to this analysis is novel and includes a large trove of original data that Code.Gov has not had until now. Nevertheless, there are some limitations to our approach.

We do not believe these limitations seriously threaten the robustness of our results or recommendations. We lay out these limitations below so that the Code.Gov team can most effectively understand and utilize our recommendations.

DATASET LIMITATIONS

Although the dataset we are working with is substantial in size and original in nature, it is not exhaustive.

We have captured data for over 5,500 repositories – over 90% of those that GitHub has labelled as belonging to federal agencies.[23] However, GSA reports that there are 8,753 federal repositories in existence (including on other platforms).[24]

We do not view this as a major impediment for our research: Many of the additional GSA-listed repositories are effectively inactive and in any case are not discoverable to members of the public.

Moreover, even if this was not the case our dataset would be sufficient for the analytical task at hand. It provides significant insight into which repositories outperform their peers. This allows us to qualitatively assess the characteristics that drive the success of top-performing repositories, and thereby produce recommendations about ways to improve other federal repositories.

IDENTIFYING CAUSALITY

We cannot conclusively establish causality using our dataset.

[21] McKinsey and Company, *Big data: The next frontier for innovation, competition, and productivity* (2011) Page 1. Available at https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Big%20data%20The%20next%20frontier%20for%20innovation/MGI_big_data_exec_summary.ashx

[22] For example, the General Services Administration maintains a dashboard on federal agency compliance with the Federal Source Code Policy. This is available at http://gsa.github.io/github-federal-stats/.

[23] We were not able to obtain data on the remaining <10% due to limitations with our GitHub scraping code.

[24] As of February 6, 2019. Data from GSA GitHub Federal Stats Dashboard, available at http://gsa.github.io/github-federal-stats/#tabs-1.

A key question in our quantitative analysis is whether the Pilot Program led to an increase in the amount of engagement with federal repositories. To do this, we track the number of engagements with federal repositories over time. However, correlation does not mean causation – we cannot conclude that changes in total engagement following August 2016 were the result of the Pilot Program alone.

We have tried to control for this by basing our analyses on a comparison between the two years prior to the launch of the Pilot Program and the two years after it. This creates a sort of 'counterfactual.' It is imperfect, because the increasing 'tech-savviness' of government means that the two periods did not have identical environmental considerations.

Nevertheless, we contend that the environment for digital government in 2014-16 and 2016-18 were reasonably similar. This means that while total causality cannot be established, approximate conclusions about the Pilot Program's effect on total engagement can be made.

INTERNAL ENGAGEMENT

Our database does not contain information on which users were responsible for which engagements.

This means that some portion of the engagements we count likely come from other federal employees, including those within the teams responsible for a particular repository. For example, two federal employees responsible for an agency repository might use GitHub functions like starring and pull requests to work on projects ("intra-agency' engagement"). Our data would count these interactions. It would also include federal employees from other agencies who engage with that repository ("inter-agency engagement").

We do not view this as a major impediment to our analysis. First, while we have no data quantifying the share of engagement that is intra-agency, anecdotal evidence gathered through the interviews we carried out for our qualitative analysis suggests it is not a large share. Second, inter-agency engagement is consistent with the Federal Source Code Policy's mission to share federal source code more widely. It is less of an issue if this is included in the counts of total engagement. Moreover, anecdotal evidence gathered through our interviews also suggests this is not the majority of total engagement with federal repositories.

QUALITATIVE ANALYSIS

After identifying the top-performing repositories, we explore the reasons for their success.

The objective is to identify best practices in repository management that could lead to increased public engagement with federal source code. To do this, we conducted interviews with 10 subject matter experts from the public, private, and non-profit sectors; ran 2 focus groups with a dozen federal employees; and reviewed the academic and industry literature.

We aggregate our findings into a new framework of nine best practices for achieving user engagement – the 'DREAM CODE' framework.

Our findings were iterated and confirmed through subsequent interviews and focus groups. We were unable to conduct a multivariate regression to statistically test our framework because we do not have the technical expertise to evaluate, for example, the quality of code that federal agencies have shared.

CONSTRAINTS ON RECOMMENDATIONS

There are several important questions that our research consciously does not answer.

These are outlined in Chapter 5 (*Recommendations*). They include questions such as “Who are the types of users engaging with federal repositories?” and “To what extent is the type of license used for a repository correlated with engagement?”

The open questions are the result of constraints that we face in our role as external researchers. These constraints include the facts that:

We are neither citizens of the US nor federal government employees. This restricted our access to data and interviews focusing on the within-agency and inter-agency aspects of the Federal Source Code Policy.

This restriction manifests in several capacities. First, it informed our decision to focus only on the second pillar of the Federal Source Code Policy. Second, it means that our recommendations around institutional changes – i.e. what resources should be provided to support open source in government and the organizational changes needed to deliver it – are brief. This is because we were unable to spend significant time talking to agencies about their internal processes.

We are not professional coders. The authors of this report have professional experience in technology and its use in the public sector. However, neither of us are programmers. This has implications for some of the findings in Chapter 4 (*Qualitative Analysis*). For example, a finding in that section is that the ‘modularity’ of code – that is, how easily components can be used independently – affect the community’s willingness to engage with it. We accepted that this relationship exists based on multiple expert interviews and an extensive review of the literature.

Nevertheless, the fact that we are not coders meant we were unable to validate it empirically. Ideally, we would have analyzed a random sample of repositories from each quintile of our repository database and tested the hypothesis that lower-performing repositories had were less modular. We do not have the expertise to judge what code is modular and what is not, however.

We are collecting data from GitHub’s external API. As has been discussed, we worked with Code.Gov to develop a custom scraping tool that could pull data from GitHub about the performance of federal repositories. However, this tool was not able to pull data on the users engaging with these repositories. This is partly due to restrictions in the GitHub API (which we believe is in place, partly, to manage privacy concerns).

The implication of this is that we were unable to conduct an analysis of the type of users engaging with federal repositories. The policy implications of a user base primarily consisting of expert coders (e.g. contractors who are trying to build relationships with government) are vastly different to those of a user base that includes a large swathe of amateur coders or organizations pulling federal projects for their own internal use.

It is important to flag these limitations upfront and contextualize the research that follows. While our findings are substantial and robust, they raise additional questions that we were unable to answer in this project. In other words, through this project we have generated a number of ‘known unknowns.’ We list these in Chapter 5 (*Recommendations*) and outline research methods that Code.Gov may be able to use in order to generate answers on them.

***Table 1*: Overview of dataset**

No. federal agencies	23
No. sub-agencies and organizations	130
No. repositories	5,672
Time period covered	Dec 16, 2009 to Jan 26, 2019
Total no. engagements	191,719
No. forks	*58,259*
No. issues	*23,455*
No. pull requests	*27,176*
No. stars	*82,829*

3. QUANTITATIVE ANALYSIS

The Pilot Program has had mixed success since its inception in August 2016. Our findings show that, at the aggregate level, there was no significant increase in the rate of engagement with federal repositories. However, there are a small number of repositories which performed very strongly and received substantial engagement. This suggests that some agencies and some projects have 'figured out' how public open source projects can be effectively initiated and managed.

In this Chapter, we pose six questions that are critical in understanding the effectiveness of the Pilot Program to date and which will be central to Code.Gov's decision-making about how to move forward. We developed these questions in consultation with Code.Gov.

Box 1

SUMMARY OF QUANTITATIVE FINDINGS

We used our dataset to answer six questions. These questions are critical in understanding the effectiveness of the Pilot Program to date and, we believe, are central to Code.Gov's decision-making in designing the future of the policy.

i. **Did the Federal Source Code Policy increase the total number of repositories created?**

   The Pilot Program did not accelerate growth in the number of publicly-available open source projects. The number of repositories grew at 5% per month in the two years preceding August 2016, compared with 2% in the following two years.

ii. **Did the Federal Source Code Policy lead to increased engagement with federal agency repositories?**

   The Pilot Program did not increase the rate of engagement with federal repositories. Total engagement grew at 8% in the two years preceding August 2016, compared with 4% in the following two years.

iii. **What is the distribution of engagement by repository?**

   A small number of high-engagement repositories drove a substantial portion of engagement. The 20 most-engaged with repositories accounted for over 40% of engagement.

iv. **Which repositories do the public most engage with? Which agencies over-perform?**

   A small number of agencies are responsible for the highest-performing repositories. NASA and the DOD contribute the most repositories to the top 20 highest-performing repositories.

v. **How does the public engage with these repositories?**

   Stars and forks count for most engagement. They were worth 54% and 40% of engagements respectively.

vi. **How responsive are agencies to engagement?**

   Agencies are somewhat responsive to engagement. On average, 96% of pull requests were acted upon and 63% of issues were closed. However, there was substantial variation amongst agencies on these metrics.

I. DID THE FEDERAL SOURCE CODE POLICY INCREASE THE NUMBER OF REPOSITORIES?

It is unclear whether the policy had an impact: While the number of repositories increased after the introduction of the Pilot Program, the rate at which they increased did not.

As Figure 4 shows, the cumulative total number of repositories increased from ~3,000 repositories in August 2016 to ~5,000 two years later. This implies that the average number of new repositories per month increased from 85 in the two years to August 2016 to 88 in the two years following it.

However, the growth rate of total federal repositories fell after the Pilot's introduction. Total repositories grew at an average of 5% per month in the two years to August 2016, compared to 2% in in the subsequent period.

It is possible that the growth rate would have tapered off even more had the Pilot Program not been in place. For that reason, we cannot conclude that it had no effect on the number of repositories. We can, however, say that the Pilot Program failed to produce a substantial or sustained increase in the growth rate. At best, it led to a continuation of pre-existing growth which may not have otherwise occurred – however, it was not a boost to growth.

II. DID THE POLICY INCREASE ENGAGEMENT WITH FEDERAL REPOSITORIES?

There was no sustained increase in engagement with federal repositories.

Engagement jumped when the Policy was announced (Figure 5). This spike was driven by stars and forks – which is encouraging in terms of the Pilot Program's objectives, since stars and forks imply that a user either has interest in following the development of federal source code or in using that code themselves.

However, the spike only lasted a month. In the two years following the policy's announcement, total engagements grew at an average of 4% per month, compared to 8% in the two years prior to August 2016. As in the previous analysis, it is possible that growth would have slowed even further had the Pilot Program not been introduced. This means the strongest conclusion we can make is that the Pilot Program did not lead to a sustained increase in the rate of engagement with federal repositories.

III. WHAT IS THE DISTRIBUTION OF ENGAGEMENT BY REPOSITORY?

Engagement with federal repositories is very skewed.

As Table 2 shows, a small number of repositories counts for most public engagement with federal repositories – specifically, 1% of federal repositories account for 51% of engagement. The mean of this distribution is 34 engagements per repository over the total period, and the median is 6.

The implication of this is that a small number of repositories significantly outperformed most federal repositories hosted on GitHub.

Table 2: Total engagements per repository

For entire period, 2008-2019

No. engagements	No. repositories
>10,000	1
5,001 to 10,000	1
1,001 to 5,000	14
501 to 1,000	31
101 to 500	228
51 to 100	208
11 to 50	957
2 to 10	1,986
1	577
0	1,169

IV. WHICH REPOSITORIES DO THE PUBLIC MOST ENGAGE WITH? WHICH AGENCIES OVER-PERFORM?

The top 20 repositories – which we call 'Superstar Repositories' – are worth 41% of all engagement.

The 5 top-ranking Superstar Repositories are:

1. *NASA's openmct*, which "is a next-generation mission control framework for visualization of data on desktop and mobile devices...[that] could be used as the basis for building applications for planning, operation, and analysis of any systems producing telemetry data.";
2. *DOD's Dshell*, which is an "extensible network forensic analysis framework";
3. *DOD's SIMP*, or System Integrity Management Platform, which can be used to build an organization's digital network infrastructure;
4. *GSA's data*, a collection of miscellaneous data from projects across the agency; and
5. *NASA's mct*, the original desktop version of openmct (the latter is a web application).

It is important to note that while NASA and the DOD contribute the most repositories to the top 20, they underperform when we consider their total collection of repositories. Both agencies have a 'long tail' of underperforming repositories. When ranking agencies by average engagements per repository, NASA and the DOD come 19th and 21st respectively out of 22 (Appendix C). When we consider both total engagement and average engagement GSA performs most consistently, ranking 3rd and 4th respectively on those metrics. The Department of the Interior also performs well, coming in at 4th and 6th.

A list of the top-performing repositories for each agency is provided in Table 4. Note that there is substantial variation between these repositories: NASA's best-performing repository has >39,000 engagements, whereas the National Science Foundation's has only four.

V. HOW DOES THE PUBLIC ENGAGE WITH THESE REPOSITORIES?

Across all repositories, most engagement comes in the form of Stars and Forks.

In our sample, Stars accounted for 54% of engagements, Forks 40%, Issues 4%, and Pull Requests 2%. This is encouraging: As already discussed, Stars and Forks suggest users want to use federally-released code for their own purposes. By contrast, Issues and Pull Requests reflect potential problems and proposed modifications to data.

That Stars and Forks account for most of the total engagement could be the result of two possibilities: Either code being shared is relatively bug-free, or the public is insufficiently engaged to bother spending time looking for problems. With a dataset this large, the truth is likely a combination of the two. We explore the implications of these two possibilities in Chapter 4.

VI. HOW RESPONSIVE ARE AGENCIES TO ENGAGEMENT?

Agencies are relatively responsive to engagement.

We measure responsiveness in two ways. In the first instance, we calculate the percentage of pull requests that have been either closed or merged, which means the suggested changes from the user have been either rejected or taken on board. On this metric, the results are strong: Across repositories, an average of 96% of pull requests are acted upon (Appendix D). Moreover, the range between agencies is relatively small: Three agencies have acted on 100% of Pull Requests, and the lowest performer, the Department of Education, has still responded to 89%. However, it is worth noting that Pull Requests – more than any other engagement we measure – are likely to come from internal sources.

Performance using the second measure of responsiveness is more mixed. Here, we calculate the percentage of Issues raised that have been closed. This means that the repository owner has addressed the potential problem raised by the user. On average, agencies closed 63% of Issues raised. The Office of Personnel Management leads with 91% and the Treasury performs worst, having only closed 37% of Issues.

One explanation for the variation may be the different amounts of initiative that pull requests and issues require of the repository owner. In pull requests, the user suggests changes for the code; in issues, they merely flag an issue. It is therefore possible that the repository owners are more willing to act upon a suggested fix than spend the time coming up with one themselves.

Figure 4

Total number of federal repositories

Cumulative total

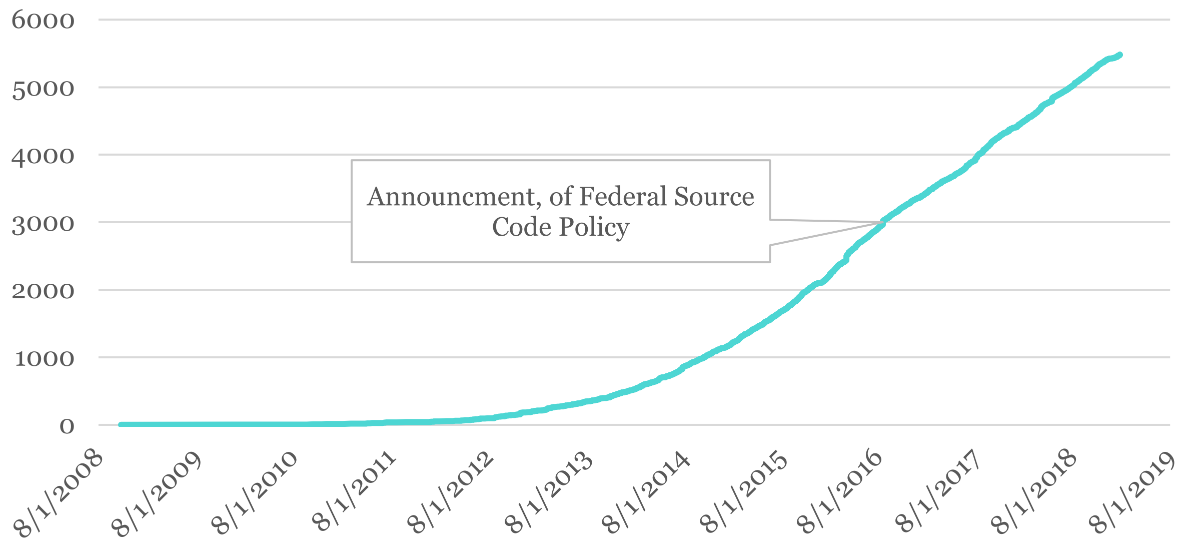


Figure 5

Total engagement with federal repositories

Cumulative total

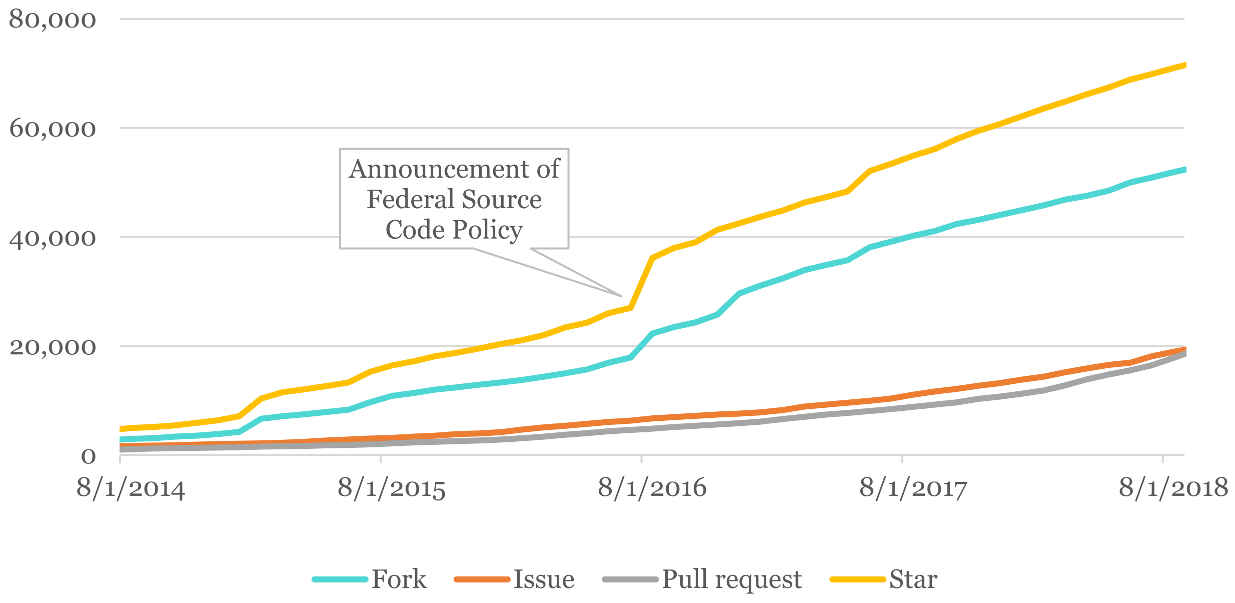


Table 3: **Repositories, ranked by total number of engagements from October 2008 to January 2019**[25]

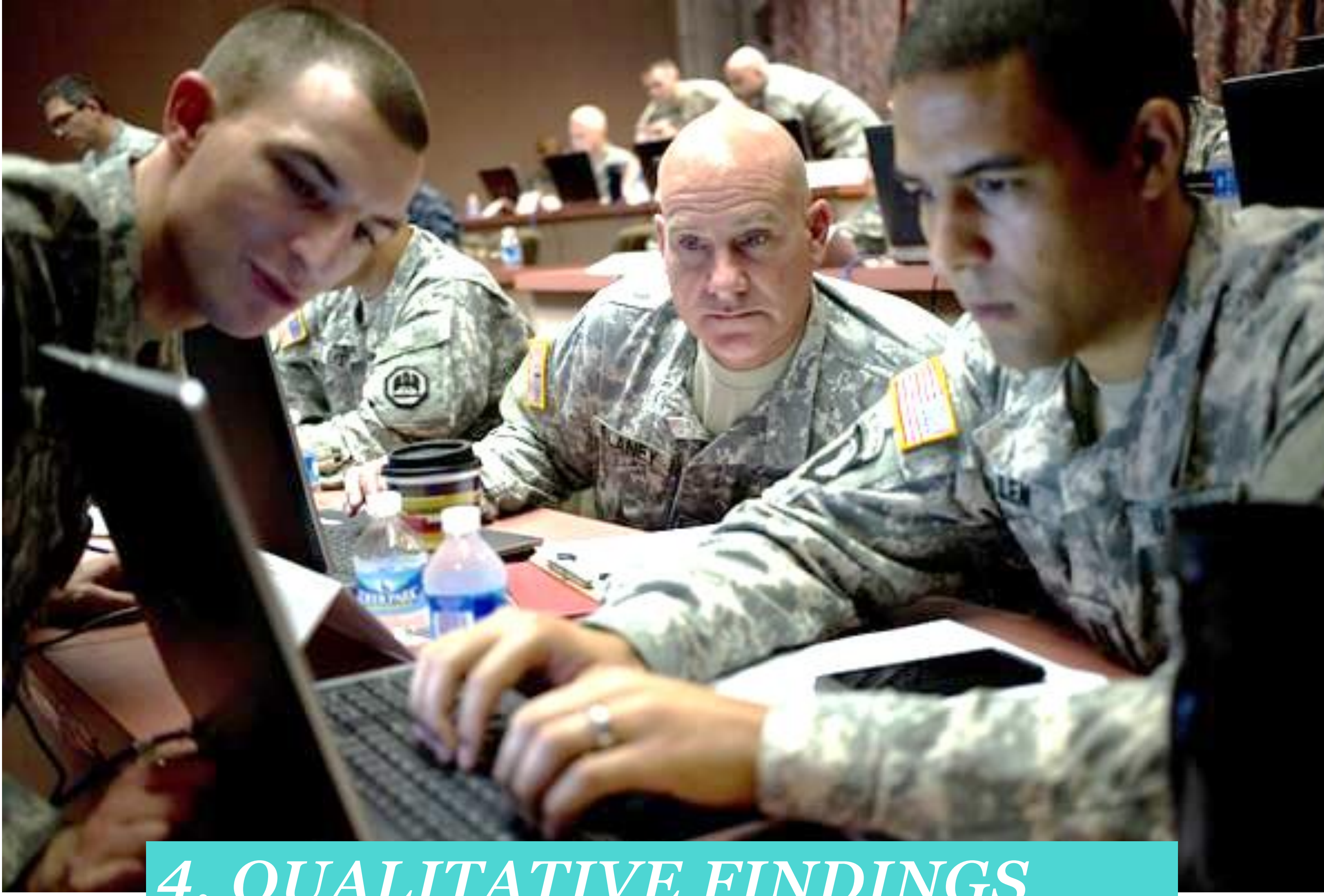
Rank	**Repository**	**Agency**	**Date created**	**Total engagements**	***Stars***	***Forks***	***Issues***	***Pull requests***
1	openmct	NASA	6/2/2015	39,731	22,011	17,612	29	79
2	Dshell	Department of Defense	12/17/2014	9,706	5,131	4,476	23	76
3	SIMP	Department of Defense	4/28/2015	4,164	2,436	1,722	-	6
4	data	General Services Administration	12/16/2014	2,330	1,379	940	1	10
5	mct	NASA	5/1/2012	2,928	1,648	1,280	111	4
6	Data.gov	General Services Administration	07/16/2013	2,312	1,304	1,002	6	-
7	NASA-3D-Resources	NASA	7/23/2014	2,288	1,480	808	9	1
8	project-open-data.github.io	Department of Commerce	1/29/2015	2,221	829	778	256	358
9	Windows-Secure-Host-Baseline	Department of Defense	2/26/2016	1,689	984	640	59	6
10	worldview	NASA	4/22/2014	1,411	360	198	518	335
11	lemongraph	Department of Defense	7/25/2016	1,325	862	440	7	16
12	sunpy	NASA	8/6/2011	1,303	387	296	192	428
13	WebWorldWind	NASA	8/29/2015	1,280	370	272	419	219
14	xAPI-Spec	Department of Defense	12/19/2012	1,262	530	590	85	57
14	goSecure	Department of Defense	4/28/2016	1,210	800	387	5	18
15	Citysdk	Department of Commerce	8/4/2016	1,136	498	270	274	94
17	team-titan	Office of Personnel Management	7/23/2018	852	-	-	852	-
18	pshtt	Department of Homeland Security	7/5/2016	843	521	141	65	116
19	WALKOFF	Department of Defense	6/8/2016	815	396	212	87	120
20	earthdata-search	NASA	8/13/2015	786	454	268	7	57
			Total	80,911	43,307	32,710	2,886	2,008

[25] Note that engagement totals will not align with the current live totals on each repository's GitHub page. The scraping tool we have used captured all engagements over time. If a user Stars something on GitHub, and later un-stars it, then that Star will appear in our totals above but not on GitHub's current live Star total for that repository.

Table 4: Top performing repositories for each agency

Agency	***Organization***	***Repository***	***Description***	***Date created***	***Total engagements***
NASA	*NASA*	*openmct*	Web-based mission control framework	6/2/2015	39,371
Department of Defense	*US Army Research Laboratory*	*Dshell*	Network forensic analysis framework	12/17/2015	9,706
Department of Health and Human Services	*Blue Button*	*Blue Button*	Personal health data application	12/21/2015	3,649
Department of Commerce	*US Census Bureau*	*Project-open-data.github.io*	Open data policy	1/29/2015	2,221
Office of Personnel Management	*USA Jobs*	*Team-titan*	Internal resource for project team	7/23/2018	852
Department of State	*Department of State*	*State-TalentMAP*	System to match State Dept employees with open jobs	5/18/2017	766
General Services Administration	*18F*	*Analytics.usa.gov*	Monitors US federal gov't web traffic	12/30/2014	750
Department of Homeland Security	*CISA*	*pshtt*	Scan domains and return data based on HTTPS best practices	7/05/2016	843
Department of Justice	*Department of Justice*	*Foia.gov*	Front end of national FOIA portal	7/10/2017	735
Department of Veterans Affairs	*Department of Veterans Affairs*	*Caseflow*	Web app to track and process appealed claims at the Board of Veterans' Appeals	2/10/2016	536
Department of the Treasury	*Federal Spending Transparency*	*Usapsending-api*	Interface to monitor public spending	8/10/2016	490

Department of the Interior	*USGS Web Informatics and Mapping*	*Whispers*	Wildlife Health Information Sharing Partnership Event Reporting System	1/11/2018	455
Environmental Protection Agency	*EPA*	*e-manifest*	Online form for hazardous waste shipments	12/1/2016	395
Department of Energy	*Lawrence Livermore National Laboratory*	*Lbann*	Artificial neural network training	5/11/2016	353
Department of Transportation	*US DOT ITS JPO ODE*	*Jpo-ode*	Interface to share connected vehicle data	10/26/2016	344
US Department of Agriculture	*Farm Service Agency*	*Fsa-style*	Style guide for the FSA	4/18/2016	284
Small Business Administration	*USSBA*	*Hubzone-webmap*	Digital map for underdeveloped hubs	10/12/2016	261
Department of Labor	*Department of Labor*	*Handbook*	Employee handbook	8/31/2016	157
Social Services Administration	*SSAgov*	*ANDI*	Tool to test web content for accessibility	8/8/2017	102
Department of Education	*Department of Education*	*Usedgov.github.io*	Developer hub for the DfE	7/13/2016	92
USAID	*USAID*	*USAID-Data-Services*	Interface to request data service support	4/3/2013	56
National Science Foundation	*National Science Foundation*	*nsf-ember-tooltip*	Tool to integrate different developer applications	6/21/2017	4



4. QUALITATIVE FINDINGS

A small number of Superstar Repositories significantly outperform their peers. The question is why. In this Chapter, we find that repositories which feature a combination of nine characteristics are most likely to receive significant engagement. These characteristics constitute the DREAM CODE framework.

There are two parts to this analysis. First, we develop the DREAM CODE framework. This was based on interviews with ten subject matter experts from the public, private, and non-profit sectors, 2 focus groups with a dozen federal employees, and a review of the academic and industry literature. In this research, we identified best practices in repository management. We then aggregated these into the DREAM CODE framework.

Box 2

SUMMARY OF QUALITATIVE FINDINGS

There are nine characteristics that drive higher levels of user engagement. These constitute the 'DREAM CODE' framework.

Discoverability: Repositories should be easily discoverable by users. This will often involve the repository owners taking a 'communication-centered' approach, for example by selecting a user-friendly name for the repository, actively linking to it on their websites, and ensuring it appears in search engine results.

Reusability: **Source code should be complete, self-contained and usable, with minimal recoding required for functional reuse**. This might also entail modularizing the code to separate out the sections that are most reusable.

End user: **Repositories should be developed with specific target populations in mind**. The population needs to have either intrinsic or extrinsic motivation to engage with the code.

Applicability elsewhere: **A repository's contents should provide a wide variety of reuse application opportunities**. They should avoid being too specialized or technical.

Maintenance: **Source code should be regularly maintained after its initial open sourcing**.

Community building: Repository owners should actively create and engage community around the project. This might be by targeting a specific community, for example by leveraging passion around a specific issue.

Open origins: **Repositories should be 'written in the open'** – that is, not built as 'closed' software and opened later. Open origins mean the code is developed with the OSS user in mind.

Documentation: **Repositories should have clear documentation and clear descriptions of their contents**. This often includes a mission statement and an outline of the repository's scope.

Explicit licensing: **Repositories should opt for an open source-friendly license** – and be explicit about the terms of that license in its read-me.

NOTES ON METHODOLOGY

Some caveats on the DREAM CODE framework and our methodology in testing it:

- Given the nature of the topic, the nine characteristics included in the DREAM CODE framework are interconnected. In some cases – depending on the specifics of the repository in question – they may overlap. For example, for some repositories, having 'open origins' and being coded in the open will likely involve regular maintenance. We have chosen to keep the nine as separate characteristics, however, in order to preserve the generalizability of the framework.

- This framework was designed with the intention of creating repositories with high engagement. Not all repositories are designed with this objective in mind – some projects, for example, may be made open so that a small number of specific contributors can work on it. As a result, not all the criteria will be relevant to all OSS projects.

THE DREAM CODE FRAMEWORK

Discoverability

Repositories should be easily discoverable by members of the OSS community.

The repository's name should be clear and relevant. As Karl Fogel, author of the leading manual on OSS practices, notes, having a "good name" is important because it's the "the first thing [a user] will encounter".[26] It should indicate the purpose of the project, be easily memorable, be distinct from other project names, and not violate any trademarks.[27]

Discoverability is particularly important in the public sector. Tee Morris, Communications Director for Code.Gov, told us that the most common problem with low-engagement repositories in his experience is names and descriptions that give no information about the repository's contents. [28] By contrast, repositories with high engagement tend to have thorough and user-friendly descriptions.

The repository's home page should also be user-friendly. This includes simple measures such as having an email address that users can contact with any questions and having an avatar for the repository owner account. [29] One subject matter expert – responsible for two of the

[26] Fogel, Karl, *Producing Open Source Software: How to Run a Successful Free Software Project*, Version 2.3098, Available online at http://producingoss.com. p. 13.

[27] Fogel pp. 13-14.

[28] Interview with Tee Morris.

[29] Interview with Eric Mill, a leading expert on open source in government with many years of experience at 18F. Mill founded Super Repository #4 and was closely involved in the development of #18.

Superstar Repositories – argues that these measures give the home page of a repository a 'human' veneer, which is vital to attracting new users.

Repository owners should take a communications-centered approach. Publicizing the repository – for example by linking to the project on agency websites, policy pages, and in email signatures – makes it more discoverable to relevant populations. [30] 18F, which has multiple repositories with high engagement, uses website footers, blogs, Twitter, external newsletters, and presentations at conferences.[31]

Reusability

Source code should be complete, self-contained, and usable. There should be minimal recoding required for functional reuse.

Code should be easy to initiate in reuse.[32] Code that can be easily automated – and made functional with limited hassle – is likely to receive high engagement levels.[33] This could be achieved via a simple command in the repository's 'Read Me' which gets the code up and running.[34]

Code should be modularized where possible. This means separating sections that are most reusable from less usable sections.[35] Research finds that when it comes to open data in government, which has many parallels with open source in government, reusability is also important for "making version management clear and reliable" and "ensuring data quality, easy-to-understand content and formatting".[36]

End user

Repositories should be designed with specific end users in mind. Owners should consider user motivations for engaging with a repository.

Owners should distinguish between the intrinsic and extrinsic motivations users have for engaging with a project. Intrinsic motivations are related to the direct potential uses of a project. Extrinsic motivations are related to the status or reputation gained by the user for engaging and contributing to a given repo. [37] Research suggests that repositories

[30] Federal employee focus group, 28 February. One participant in our federal government focus group noted that an agency making its open source repositories easily discoverable on the agency's website is typically correlated with high levels of user engagement. Another participant noted the importance of a repository appearing in search engine results and being search engine optimized (SEO) for it to attract users.

[31] Britta Gustafson, a content designer at 18F, noted this in an official public comment on behalf of 18F on GitHub. Available at: https://github.com/WhiteHouse/source-code-policy/issues/94.

[32] Interview with Ricardo Reyes. Ricardo is Open Source Director in the Code.Gov team and is responsible for community and agency outreach.

[33] Told to us by two federal employees in one of our focus groups.

[34] Interview with Eric Mill.

[35] Interview with Ricardo Reyes.

[36] Sushchenia, Iryna and Grönlund, Åke, Organizational measures to stimulate user engagement with open data, *Transforming Government: People, Process and Policy* Vol. 9 No. 2, 2015 p. 194.

[37] Roberts, Jeffrey A., Hann, Il-Horn, and Slaughter, Sandra A., Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects, *Management*

satisfying both of these motivations are more likely to get higher engagement.[38]

If reach is the objective of the repository, projects should avoid being too tightly tied to one end user.[39] Otherwise it will be too hard for other end users to use it for their own purposes.

Other projects may focus on smaller – but passionate – audiences. Projects most frequently used by the scientific community are a good example.[40]

Owners should be deliberate in which programming language they use. It may be even more important than the quality of the code itself.[41] For example, while Python is a popular OSS language, relatively few government repositories use it.[42] Some federal teams are already consciously choosing programing languages based on potential user engagement - for example, Code.Gov chose Reactive over Angular as its base language specifically because it would increase the public's ability to contribute to the project.[43]

(*Note*: Very few repos have only one language – one study of repositories on GitHub found that around 97% of projects contained two or more languages.[44] However, whether specific modules contain one language over another can affect engagement).

Applicability Elsewhere

The project should be relevant beyond its origin intent.

Owners should prioritize library code over application code.[45] The former includes projects that are specifically designed to be reusable in multiple scenarios. The latter includes projects which are created for specific purposes.

Owners should prioritize code that can be used for infrastructure and platform projects.[46] This is especially true of code that works cross-platform (e.g. on both Windows and Linux).[47] Source code should ideally also be system interoperable, meaning that it has the "ability to transfer and use information in a consistent, efficient way across multiple organizations and IT systems to accomplish operational missions".[48]

Science Vol. 52, No. 7, Open Source Software (Jul., 2006), pp. 986.

[38] Roberts et al. p. 996.

[39] Interview with private sector industry expert who spoke on condition of anonymity.

[40] Participant in federal employee focus group.

[41] McDonald, Nora, and Goggins, Sean, Performance and Participation in Open Source Software on GitHub, CHI EA '13 CHI '13 Extended Abstracts on Human Factors in Computing Systems. P.. 144.

[42] Interview with Amin Mehr from the Code.Gov team.

[43] Interview with a senior member of the Code.Gov team.

[44] Tomassetti, Federico , and Torchiano, Marco, An Empirical Assessment of Polyglot-ism in GitHub - *EASE '14*, May 13 - 14 2014.

[45] Interview with Eric Mill.

[46] Interview with David Eaves.

[47] Interview with Eric Mill.

[48] "Project Interoperability." Project Interoperability, project-interoperability.github.io/.

Maintenance

Repositories should be regularly maintained once created. This includes updates for new features, bug fixes, and other changes.

Owners should encourage diversity in their community of maintainers. Having a broader mix of individuals maintaining the code base increases the quality and regularity of maintenance.[49] For this reason, the OSI requires that its members seek "active participation from multiple contributors, i.e. individuals and organizations other than founders".[50]

One metric to assess the maintenance health of a repository is the frequency of 'commits' on GitHub.[51] Another is how recently the last meaningful commit was performed.[52]

Owners should consider continuous integration as a way of improving the quality of their maintenance process.[53] This mechanism ensures quality standards "running tests every time you push a new commit and reporting the results to a pull request."[54] Responding to pull requests swiftly also helps this effort.[55]

Provide a clear and accurate statement about the project's development status. This includes outlining for users what stage of development the software in question is in, to avoid a scenario of over-promising and ultimately repelling users through disappointment.[56]

A good maintenance history will encourage user engagement. Because it is difficult to assess the quality of a repository's code from the outset, users may often look at a repository's development history (i.e. maintenance history) as a proxy for its quality when deciding whether to engage with the code.[57]

Community building

Owners should focus on building a lively community around their project.

[49] Told to us by participant in federal employee focus group.

[50] "Affiliate Membership Qualifications and Criteria." *Affiliate Membership Qualifications and Criteria | Open Source Initiative*, opensource.org/AffiliateRequirements.

[51] Commits are individual changes to a file or set of files within a repository.

[52] 'Meaningful' here means a substantial change in the source code, rather than something more administrative, such as correcting a typo. There are a number of ways to consider 'recent.' CHAOSS, a Linux Foundation project looking at Community Health Analytics of Open Source Software, notes the varying levels of nuance that could be used when analyzing frequency of commits. For example, CHAOSS notes it could be useful to distinguish between the number of commits made and the number of commiters, or number of commits and the number of lines of code added or changed per commit. See "Metrics With Greater Utility: The Community Manager Use Case." CHAOSS, 25 Feb. 2019, chaoss.community/news/2018/11/16/metrics-with-greater-utility-the-community-manager-use-case/.

[53] This was suggested by the owner of a Superstar Repository in a focus group.

[54] This explanation of continuous integration is courtesy of Nicolai, Johannes. "GitHub Welcomes All CI Tools." The GitHub Blog, 4 Jan. 2019, github.blog/2017-11-07-github-welcomes-all-ci-tools/.

[55] Participant in federal employee focus group.

[56] Fogel p. 17.

[57] Ndenga, Malanga Kennedy, Jean, Mehat, Ganchev, Ivaylo, and Franklin, Wabwoba Assessing Quality of Open Source Software Based on Community Metrics, *International Journal of Software Engineering and Its Applications*, (2015) 9:12, 337-348.

Repository owners should view community outreach as an indispensable part of the open source process. Without a lively community, an OSS project is little more than code which happens to be shared online. The benefits of OSS that come with public engagement – for example, bug fixes and the creation of new capabilities – only exist to the extent there is a vibrant community around the code.

This is particularly a problem in the public sector. One expert described community outreach as "the major blocker to public engagement" in federal OSS projects, pointing out that many agencies have neither the time nor the resources to do this.[58]

In many cases, community outreach is currently done on the government's terms (which the expert described as an approach of "come to our agency to talk to us on how you engage with your code"). By contrast, best practice is to take proactive steps such as organizing conferences where developers can interact with federal employees and repository owners directly.

Projects should have full-time community managers attached to them. One reason for this problem is that some federal employees tasked with managing open source projects are not experts in the field. Ensuring that projects have full-time community managers helps to increase engagement because it (a) ensures alignment between the project and what the community can reasonably contribute to, and (b) provides a constant interface with the public to facilitate any public engagement.

This also mitigates the issue – common in federal government – where agencies discourage developer feedback. This is because of the perceived political and communications risks, rooted in nervousness about unfamiliar technologies.[59]

One interviewee noted that GSA and NASA – two agencies which feature overwhelmingly in the list of Superstar Repositories – have been able to mitigate this risk by opting for a "forgiveness rather than permission" workflow when interacting with developers. The alternative – a permission-based review process for any engagement with users in a given repository – can be laborious and becomes fatal to community building.

Owners should be responsive to issues posted by users on GitHub. Some agencies address issues offline. This is a basic means to promote community engagement.[60] A strong presence on other social media used by the open source community also helps.[61]

Projects should create structured forums for community engagement. For example, the manager of one Superstar Repository told us that he hosted a weekly conference call for GitHub users to discuss Pull Requests and Issues.[62] They also had a kickoff webinar when the project was originally launched, and used GitHub as a

[58] John Scott, from Ion Channel, is an open source contractor for the federal government with particular experience on Department of Defense software policy .

[59] Interview with Eric Mill.

[60] Interview with Eric Mill.

[61] Interview with Eric Mill.

[62] Federal employee focus group participant who manages a Super Repository.

way to have a one-stop shop and on-going collaboration on the project (rather than handling it through email).

Planned forums for community engagement should be documented. In its Affiliate Member requirements, OSI stipulates that organizations should have a "documented approach for participation by the public" as well as "methods for current and interested individuals/organizations to join and participate in your community".[63]

Community managers should cultivate a productive and civil tone amongst members. A repository's text and tone determine whether it has 'a human face' – and hence entices engagement.[64] Codes of conduct contribute to this tone.[65]

Open origins

Projects should be open from Day 1.

The longer a project is run in a closed source manner, the harder it is to open source later.[66] This is because the longer a repository is closed, the more likely it is to contain sensitive or confidential information and passwords within the code.

This risk is particularly acute in the public sector context: For example, sensitive government passwords or citizen data could be accidentally released in a late-stage switch to open.[67] Flipping from closed to open is hard because the agency must vet code for any security issues.[68] It is also difficult to get teams to make the cultural switch to open development.

Other public sector organizations take this approach. The UK's Government Digital Service, for example, explicitly advocates for 'coding in the open.'[69]

Documentation

Repositories should come with clear documentation that describes its contents. Documentation "is essential" because "there needs to be *something* for people to read, even if it's rudimentary and incomplete."[70]

However, it can also place a high burden on the developer. It is therefore useful to establish minimal criteria for user-friendly documentation – for example, a description of the minimum technical knowledge required of the user, information about setting up the software, an example of how to perform the basic task using the software, and acknowledgement of any deficiencies or

[63] "Affiliate Membership Qualifications and Criteria." *Affiliate Membership Qualifications and Criteria | Open Source Initiative*, opensource.org/AffiliateRequirements.

[64] Fogel p. 118.

[65] Fogel p. 28.

[66] Fogel p. 31.

[67] Fogel p. 88.

[68] Participant in federal agency focus group.

[69] "Making Source Code Open and Reusable." GOV.UK, www.gov.uk/service-manual/technology/making-source-code-open-and-reusable. For more, please see Shipman, Anna. "The Benefits of Coding in the Open." Government Digital Service, gds.blog.gov.uk/2017/09/04/the-benefits-of-coding-in-the-open/ and Shipman, Anna. "Don't Be Afraid to Code in the Open: Here's How to Do It Securely." Technology in Government, gdstechnology.blog.gov.uk/2017/09/27/dont-be-afraid-to-code-in-the-open-heres-how-to-do-it-securely/.

[70] Fogel p. 20.

missing parts within the software.[71] This helps limit the amount of work required of the project owner.

Documentation should be comprehensive but clear. Complicated language is likely to turn away community members.[72]

Wikis and Read Me files should be well developed. Shortcuts on these elements can be detrimental to community growth.

Badges should be effectively used.[73] This is a clear and simple means to signify to users the development stage and quality of the code and appropriately set user expectations.

A clear mission statement should be provided. A clear description – which briefly outlines the contents and purpose of the project – enables potential users to decide whether the repository in question suits their purposes.[74] This should be accompanied by a features and requirements list that "clarifies the mission statement's scope".[75]

EXPLICIT LICENSING

Repository owners should make explicit choices about licensing before making a project open source. They should be clear to the community about what the chosen license is.[76]

Licensing is a critical driver of user engagement: Choosing a license and clearly stating it is a central design step.[77] Sophisticated community members will often choose not to engage in projects unless there is clarity around licensing questions.[78] Many users don't want to contribute to a project that may later become a commercial entity's IP.

Government agencies often create OSS projects without thinking about these questions and without sharing their perspectives on them. This, in turn, deters potential community members.

The Army Research Lab and NASA are two examples of agencies that tend to manage licensing up-front well.[79] They generally attach permissive licenses to projects.

[71] Fogel p. 20.

[72] Told to us by the manager of a Super Repository in the federal employee focus group.

[73] Badges are a community norm on GitHub and a means "to signal to fellow developers that we set ourselves high standards for the code we write". This explanation is courtesy of GitHub repository's explanation of the value of badges https://github.com/dwyl/repo-badges

[74] Fogel p. 15.

[75] Fogel p. 16.

[76] See Fogel Chapter 9 "Legal Matters: Licenses, Copyrights, Trademarks and Patents", for example, if interested in more on this issue.

[77] Fogel p. 24.

[78] Interview with subject matter expert who spoke on condition of anonymity.

[79] Interview with subject matter expert who spoke on condition of anonymity.

5. RECOMMENDATIONS

The Pilot Program has had some successes – but there is more work to be done. The analyses in Chapters 3 and 4 show that engagement with federal source code is inconsistent amongst agencies and projects. This implies that there is some sort of inefficiency in the policy. As the August 2019 Pilot Program deadline approaches, it is up to Code.Gov and GSA to determine what changes – if any – ought to be made to fix this.

In this Chapter, we outline major decisions that need to be made in the run-up to August 2019 and the design of 'Open Source Policy 2.0.' We also provide our recommendations for which alternatives should be chosen in these decisions. Specifically, we recommend that a redesigned policy will need to make choices at four decision nodes:[80]

- Articulate the policy's purpose clearly;
- Amend the requirement that 20% of federal source code be released publicly;
- Explore further programmatic infrastructure that can support agencies in their efforts to release open source code; and
- Commit resources to research and analysis of open questions raised by this research.

[80] Note that this Chapter assumes that discontinuing open source in the federal government is a non-starter for Code.Gov. The brief from our client was to assess the effectiveness of the Pilot Program and suggest improvements that could help increase this. Larger decisions about whether to continue the Pilot Program or not rest on political considerations that are beyond the remit of our research.

I. DEFINE THE OBJECTIVE OF FEDERAL OSS

When the Obama Administration announced the Federal Source Code Policy in August 2016, several justifications for the policy were offered:

1. **Generating procurement savings:** Creating cost savings and improve procurement practices, by avoiding "duplicative custom software purchases" amongst agencies and vendor lock-in;
2. **Encouraging innovation within government:** Fueling transparency, innovation and better software engineering in government, by promoting "collaboration across Federal agencies" on projects;
3. **Sparking contributions from beyond government:** Facilitating qualitative improvement to federal source code, by enabling members of the public to help develop code that's "reliable and effective in furthering our national objectives;"[81] and
4. **Enabling third party reuse:** Honoring public ownership of the code by providing the public with "the People's code" that their taxpayer dollars fund.[82]

Objectives 1 and 2 are primarily addressed by the first pillar of the Policy, which is aimed at encouraging interagency coordination and sharing on source code.

Objectives 3 and 4 are notionally tied to the second pillar, i.e. the Pilot Program. The policy states that agencies "should develop and release the code in a manner that fosters communities around shared challenges and improves the ability of the OSS community to provide feedback on, and make contributions to, the source code"[83] (i.e. Objective 3). It also states that "when deciding which custom-developed code projects to release, each agency should prioritize the release of custom-developed code that it considers potentially useful to the broader community" (i.e. Objective 4).

This report has focused on the question of whether the Pilot Program has met Objective 4 – that is, whether the public has in fact used the source code released in the Pilot Program.

In practice, however, it is not clear whether the federal government's Open Source Pilot Program in its implementation has been primarily intended to solicit contributions from the public (Objective 3), to share OSS with the public (Objective 4), or some combination of the two. Multiple interviewees questioned whether public-facing elements of the Federal Source Code Policy are – or should be – a priority for open source in government. They argued that the procurement and innovation rationales (Objectives 1 and 2) are in fact the major motivation for the federal government's use of open source. As one commentator has written, "the federal source code policy is decidedly *not* an open-source policy. Rather, the policy was principally directed at government-wide reuse of source code and making sure that agencies could find other agencies' source code."[84]

[81] Scott, Tony, U.S. Chief Information Officer. "The People's Code." *National Archives and Records Administration*, National Archives and Records Administration, 2016, obamawhitehouse.archives.gov/blog/2016/08/08/peoples-code.

[82] Scott, Tony, U.S. Chief Information Officer. "The People's Code." *National Archives and Records Administration*, National Archives and Records Administration, 2016, obamawhitehouse.archives.gov/blog/2016/08/08/peoples-code.

[83] "Federal Source Code Policy." Open Source Software, sourcecode.cio.gov/OSS/.

[84] Zvenyach, V. David. "The Trouble with the Federal Source Code Policy, and What to Do about It: Part One." *Medium*, 9 Oct. 2018, medium.com/@vdavez/the-trouble-with-the-federal-source-code-policy-and-what-to-do-about-it-part-

The lack of clarity on this point matters: It means that different agencies have interpreted and applied the policy differently, and that by trying to achieve multiple goals with limited resources, the Pilot Program has struggled to achieve strong results in any of them. This is apparent in the findings from Chapter 3. While the Superstar Repositories garnered significant engagement – meeting Objectives 3 and 4 – the 'long tail' of repositories with low engagement were neither reused nor contributed to by the public. This means the federal government is not living up to its stated ambition of providing the public with "useful" code.[85]

We do not provide a specific recommendation about what objective Open Source Policy 2.0 should pursue. That is a political decision that is subject to the priorities of the current administration. However, in the remainder of this chapter we do assume that Code.Gov and GSA will continue to seek public engagement with federal source code. That means we assume that Open Source Policy 2.0 will intentionally prioritize Objective 3 or 4.

Once the Code.Gov team has determined its objectives of an open source policy in Stage I, there are three decisions that it needs to take in designing the next iteration of the policy.

Note that given the scope and focus of this research on user engagement with the federal government's source code, the decisions and alternatives presented are focused predominantly on user-engagement related elements of the policy. Should Code.Gov decide to focus on Objectives 1 and/or 2 (Government Procurement and Innovation) rather than Objectives 3 and/or 4 (User Contributions and Reuse), Code.Gov needs to undertake further work to analyze the best way forward to achieve these objectives.

II. AMEND THE "20% REQUIREMENT"

Code.Gov should reconsider the requirement that 20% of federal source code be released publicly.

A recurring discovery of our research is that the Second Pillar's "20% mandate" – the guideline that federal agencies should release that amount of their source code publicly – is flawed. It is not clear what metric agencies should use to measure 20% of their code.[86] It could mean, for example, 20% of total projects, of SLOCs, or of costs.

The original intention was that Code.Gov would provide guidance around how to measure the 20% requirement. The policy states that:

"Agencies should calculate the percentage of source code released using a consistent measure—such as real or estimated lines of code, number of self-contained modules, or cost—that meets the intended objectives of this requirement. Additional information regarding how best to measure source code will be provided on Code.gov."[87]

In practice, agencies are still able to define their own metric. Code.Gov advises agencies that "[h]aving established an inventory of

one-f1f26d0232ab. Zvenyach served as a Senior Technical Advisor and Assistant Commissioner for the US General Service Administration's Federal Acquisition Service Office of Systems Management and as Executive Director of 18F [information taken from his official website at https://esq.io/pages/about.html].

[85] Section 5.1 of the Federal Source Code Policy states that agencies "should prioritize the release of custom-developed code that it considers potentially useful to the broader community". "Federal Source Code Policy." Open Source Software, sourcecode.cio.gov/OSS/.

[86] This was a criticism the policy came under from its initial drafts that were released for public comment See, for example, 18F's public comment on the issue at https://github.com/WhiteHouse/source-code-policy/issues/179.

[87] "Federal Source Code Policy." *Open Source Software*, sourcecode.cio.gov/OSS/.

new custom-developed code, agencies will still need to determine their method of measuring the amount of code it represents."[88] They then provide some options that agencies can choose from, including those mentioned above.

The result is that the amount of code which agencies release as open source varies significantly.[89] Moreover, the source code which is released is often made public simply to meet the 20% target, rather than to promote any of the objectives outlined earlier in this Chapter.

The choice of 20% as a target seems to have been the result of political considerations, rather than a deliberate choice to maximize the effectiveness of the policy. Multiple interviewees said that the 20% figure was chosen as a way to balance a faction arguing for 100% OSS on the one hand and a faction arguing for 0% on the other. It was also chosen, they said, to maintain the federal government's commitment to technology neutrality.[90]

EVALUATION METHODOLOGY

For the following set of policy design decisions, we use a simple evaluation methodology to recommend options.

We measure each option against the potential objectives of an Open Source Policy 2.0. We use a simple scoring system: the option receives a 1 if it is expected to positively contribute to that objective, a 0 if it will likely have no effect, and a -1 if it is expected to actively detract from that objective.

We then add the totals to produce a recommendation. Note that we assume Code.Gov will decide that Open Source Policy 2.0 should continue to promote user engagement. This means the ultimate set of recommendations are designed to encourage engagement. However, if Code.Gov decides that the procurement and transparency objectives will be the cornerstone of any future policy, these recommendations would change.

OPTIONS

Open Source Policy 2.0 needs to address the 20% ambiguity. This could be done in several ways:

1. **Mandate a 'default to open source' approach ("100% requirement").** All federal source code should be made open except in particular cases (for example, where there are security concerns).
   This approach is used by other governments, including the UK.[91] It is also supported by many in industry.[92]

   Benefits: Enforcement would be much easier, since it is easier to identify agencies that are not sharing

[88] "Measuring Source Code", *Code.Gov*, https://code.gov/about/open-source/measuring-code.

[89] GSA hosts a dashboard measuring agency compliance with the Federal Source Code Policy. It shows that only 3 agencies are fully compliant with the Policy. Available at https://gsa.github.io/compliance-dashboard-web-component/.

[90] The federal government's stance is that "agencies must consider open source, mixed source, and proprietary software solutions equally and on a level playing field, and free of preconceived preferences based on how the technology is developed, licensed, or distributed." "Three-Step Software Solutions Analysis." *Three-Step Software Solutions Analysis*, policy.cio.gov/source-code/three-step-software-solutions-analysis/.

[91] Point 8 of the UK government's Digital Service Standard requires this. Please see Government Digital Service. "8. Make All New Source Code Open." *GOV.UK*, GOV.UK, 29 June 2016, www.gov.uk/service-manual/service-standard/make-all-new-source-code-open.

[92] For example, Zvenyach, V. David. "The Trouble with the Federal Source Code Policy, and What to Do about It: Part One." *Medium*, 9 Oct. 2018, medium.com/@vdavez/the-trouble-with-the-federal-source-code-policy-and-what-to-do-about-it-part-one-f1f26d0232ab.

all of their code than it is to identify agencies sharing >20% of code.

Moreover, there would be more opportunities for the developer community to contribute to federal code. The net effect would be greater innovation in federal code.

Costs: The major problem is political feasibility. Our interviews revealed strong resistance from many agency representatives towards this approach, given the administrative burden it would put on them.

2. **Introduce a set of criteria that determine what projects should be open sourced ("0% with exceptions requirement")**. Instead of using a volumetric requirement, as the Pilot Program has done, Code.Gov could instead provide a 'checklist' that defines what code should be made open source. The DREAM CODE framework is one example: Assuming the goal of the policy is to stimulate user reuse, then any federal code that displays X of the 9 DREAM CODE characteristics would be required to be made open source.
   Benefits: The projects made public would be those benefiting most from the open source community.

   This would also stop resources being wasted on sharing projects that are unlikely to benefit from being open.

   Costs: Providing a 'check list' of specific characteristics provides multiple opportunities to argue that a project should not be made open. If an agency already views the use of open source as an administrative burden, this approach would make it easier to argue that projects should remain closed.

   Even those agencies that actively support OSS efforts would suffer from the increased administrative burden.

   Finally, it would be difficult to enforce: Code.Gov would have to audit closed projects according to a complicated checklist, which would be a timely endeavor.

3. **Maintain the 20% requirement but provide a uniform definition of the metric**. Code.Gov would provide a single way of measuring the 20% requirement.

   Benefits: This would make enforcement easier, since Code.Gov would assess compliance against a single standard rather than whatever the federal agencies choose to use.

   Costs: This would not address many of the problems that the 20% figure creates. For example, agencies could still release code that does not encourage public engagement.

4. **Continue with the status quo by maintaining the 20% requirement as is.**

   Benefits: This is highly politically feasible, given it is the status quo. Moreover, there is no 'learning curve' in implementing a new policy.

   Costs: Our research has shown this approach is inefficient and ineffective. It doesn't optimize for any specific policy goal.

   In addition, the volumetric requirement is a process burden – agencies need to define a way of measuring 20% and monitor their compliance with it – which creates complexity for federal employees and developers.

RECOMMENDATION

Code.Gov should default to open by setting a "100% requirement (with exceptions)."

Option A would have a positive impact on all four potential objectives for the policy (Table 6). It strictly dominates Option B, "Qualitative Criteria," which is not expected to have a significant impact on either government procurement or innovation within government. The other two options are not expected to have significant impacts on any of the four objectives. Regardless of[93] which objective Code.Gov lands on for Open Source Policy 2.0, defaulting to open is the recommended option.

Although this may be less politically feasible than other options, the fact that other governments – including the UK – have done this show it is possible in the public sector.

[93] That is, regardless of which objective Code.Gov decides to prioritize, the option would be recommended.

Table 6: Options for volumetric decision node

Option	Objectives				
	1. Gov't procurement	*2. Innovation within government*	*3. Innovation from the public*	*4. Third-party reuse*	*Total*
A. Default to Open	1	1	1	1	**4**
B. Qualitative Criteria for Open Sourcing	0	0	1	1	**2**
C. Maintain 20% requirement, with pre-defined metrics	0	0	0	0	**0**
D. Maintain 20% requirement, with agency freedom to define metric	0	0	0	0	**0**

Table 7: Options for programmatic support decision node

Option	Objectives				
	1. Gov't procurement	*2. Innovation within government*	*3. Innovation from the public*	*4. Third-party reuse*	*Total*
A. Employ community managers	0	0	1	1	**2**
B. Provide community-building training to agencies	0	0	1	1	**2**
C. Train government acquisition employees	1	1	1	1	**4**
D. Convene Federal CIO Council	1	1	1	1	**4**
E. Introduce incentives to federal employees to engage in open source	0	0	1	1	**2**
F. Publish rankings on agency performance on user engagement	0	0	1	1	**2**
G. Create 'OSS Parachute Team'	1	1	1	1	**4**

III. PROVIDE PROGRAMMATIC SUPPORT FOR AGENCIES

Federal agencies vary significantly in their capacity to comply with the Federal Source Code Policy effectively.

Our focus groups with federal employees and interviews with subject experts revealed that some agencies have substantial in-house open source expertise – particularly science-related agencies, like NASA and the DOD. Others, however, do not have large institutional understanding of open source.

This matters. One example of how it manifests is in licensing. Contributors to the open source community often elect to work on a project only if its license ensures it will not be later sold commercially.[94] However, a number of federal government projects are released under licenses that would allow for later commercialization. [95] The reason, according to several interviewees, is that procurement functions do not realize the downstream effects of licensing decisions.[96] The result is that engagement is lower than if alternative licensing had been used.

The interagency component of the Federal Source Code Policy is beyond the remit of this research, because we are neither US citizens nor federal employees. However, in the course of our research we have consistently heard that changes to the code requirements – for example, adjusting the 20% requirement or introducing the DREAM CODE framework – will not alone be enough to boost engagement with federal source code. As John Scott told us, "it's a long-term organizational issue – culture precedes problems."[97]

This problem is not unique to the Federal Source Code Policy. As one commentator on the federal government's OSS policy, who served as a Senior Technical Advisor and Assistant Commissioner for the US General Service Administration's Federal Acquisition Service Office of Systems Management and as Executive Director of 18F, writes,

"At some level, this [low] level of compliance should be expected for any top-down policy. The reality on the ground is that agencies almost always struggle to implement government-wide policies. The incentives are rarely aligned, the practices and culture needed to support the policies typically does not have the level of focus required, and things just take time (particularly where there are multiple competing priorities and pressures and leadership change). Institutional inertia is a barrier that any OMB policy must grapple with."[98]

Refining the objective of the Federal Source Code Policy and changing the 20% requirement are alone unlikely to resolve these underlying issues. The purpose of this section is to provide illustrations of the sort of additional initiatives that Code.Gov would likely need to launch as part of an Open Source Policy 2.0. These policy decisions are beyond the scope of our original research briefing from Code.Gov – in the following section, however, we explain why we have included them as part of our recommendations.

[94] Interview with subject matter expert who spoke on condition of anonymity.

[95] Interview with subject matter expert who spoke on condition of anonymity.

[96] Interview with subject matter expert who spoke on condition of anonymity.

[97] Interview with John Scott.

[98] Zvenyach, V. David. "The Trouble with the Federal Source Code Policy, and What to Do about It: Part One." *Medium*, 9 Oct. 2018, medium.com/@vdavez/the-trouble-with-the-federal-source-code-policy-and-what-to-do-about-it-part-one-f1f26d0232ab.

OPTIONS

Code.Gov should provide new forms of support to help agencies comply with Open Source Policy 2.0.

These options are drawn from suggestions made by our expert interviewees, focus group participants, and in the literature. They are, in most cases, not mutually exclusive. Moreover, they do not form an exhaustive list of the forms of programmatic support that Code.Gov could provide to agencies in order to boost user engagement with federal repositories.

Nevertheless, we have included these policy options in order to illustrate the diversity of initiatives that Code.Gov could take beyond re-writing the Federal Source Code Policy's purpose and volume requirement. Multiple interviewees said that even if those two levers were changed, many federal agencies will still lack the capabilities and the expertise to increase public engagement.[99]

This section is included in the report in order to spark thought about what additional support Code.Gov can provide. For this reason, the options below are neither mutually exclusive nor exhaustive. In the recommendation section, we show how these initiatives could be combined in different ways depending on the overarching purpose for Open Source Policy 2.0 that Code.Gov decides on.

Initiatives that Code.Gov could launch in order to boost agencies' OSS capabilities and expertise include the following:

1. ***Mandate that all federal agencies employ full-time community managers***. It is common in private sector organizations to have a staff member that manages engagement with a project's community. This includes everything from setting the community rules at the start of project to organizing conferences during the project's development, where community members can engage with the federal government. These are common practices in private sector organizations that use open source well.[100]

   Benefits: This would likely be highly effective in generating user engagement. It was one of the most popular suggestions that appeared in our research.

   Costs: There would be a considerable financial cost if each of the >5,000 federal repositories had their own community member. Even if each community manager had 100 repositories to oversee, that would still implies 50 new federal hires.

2. ***Provide community-building training to agencies.*** This could entail some combination of workshops and online modules run by experts at Code.Gov.

   It would address the problem that many agencies do not have sufficient internal expertise on how to engage communities.

   Benefits: Less costly than hiring full-time community engagement managers – while still addressing agencies' general lack of competency in community-building.

   Costs: Less effective than hiring full-time community engagement managers. It would also entail financial and administrative costs for Code.Gov, if they were to manage this effort.

3. ***Train government acquisition employees in 'Open Source 101'***. A consistent finding in our research is that many staff in agencies'

[99] Eric Mill was one example of an interviewee who said this.

[100] Interview with John Scott.

procurement arms are not well-versed in open source.

One implication of this is that the wrong licenses are used for projects, as mentioned earlier.[101]

Benefits: Increases the likelihood that OSS solutions are used to meet federal requirements (i.e., federal acquisition employees are less likely to issue an RfP for a proprietary solution when a potential open source solution exists).

This measure would also mean that more projects use the appropriate licensing arrangements from the outset, which increases engagement over the long-run.

Costs: GSA and/or Code.Gov would bear the administrative and financial burden of running these training programs. It is also not clear how effective trainings will be; OSS is a complicated domain and agencies may require deeper expertise in the field than workshops can provide.

4. ***Convene the Federal CIO Council to lead the design process for Open Source Policy 2.0***.

Since Code.Gov moved to GSA from the White House, there is a perception in government and industry that OSS has fallen off the federal government's radar.[102]

Convening the CIO Council in the run-up to the August 2019 deadline would increase the possibility that resources would be devoted to exploring institutional ways to improve the efficacy of the Federal Source Code Policy.

Benefits: Incurs no direct financial cost – and, if CIOs engage with the topic, could bring high-ranking interest and resources to the redesign process.

Costs: Low-to-moderate political feasibility, given the current Administration's relative lack of expressed interest in OSS compared to other issues in their remit, such as cybersecurity.

Code.Gov does not have the power to convene the Federal CIO Council.

5. ***Introduce incentives for federal employees to contribute to open source projects***.

Research suggests that organizations which encourage employees to spend time working on external open source projects improve the productivity of their own open source projects.[103] For agencies with fewer experts in open source, this may be one way to boost the skills of federal employees managing projects.

One way to do this is to permit certain federal employees to allocate time – say, several hours a week – to working on non-government OSS projects.

Benefits: Increases in-house OSS expertise without incurring any direct financial cost.

Costs: Consumes time from federal employees' days (which could be

[101] Interview with John Scott

[102] Interview with industry expert.

[103] Nagle, Frank, Learning by Contributing: Gaining Competitive Advantage Through Contribution to Crowdsourced Public Goods, *Organization Science*, *Organization Science*, 2018, Vol.29(4), p.569-587 and Senz, Kristen, The Hidden Benefit of Giving Back to Open Source Software, *HBS Working Knowledge*, September 5 2018, https://hbswk.hbs.edu/item/the-hidden-benefit-of-giving-back-to-open-source-software.

spent working directly on federal OSS projects).

While research shows that this is a good way to develop OSS expertise, the aggregate amount of time devoted to this initiative could exceed the cost of hiring full-time OSS experts within agencies.

6. ***Publish rankings of agency performance in user engagement in open source.***

   A study of open source across Spanish city governments found that publishing rankings of cities' performance boosted productivity and community building.[104] A similar interagency approach could have similar results.

   Currently, GSA has a public dashboard that evaluates agencies' compliance with the Federal Source Code Policy. However, this dashboard does not measure user engagement and it is relatively hard to find online. Moreover, agencies are not ranked against one another.

   Adjusting this evaluation process so that it directly compares agencies based on user engagement outcomes could provoke healthy competition which results in increased user engagement.

   Benefits: No direct financial cost.

   Costs: This intervention would likely be less effective than others. It would also be an added administrative burden for Code.Gov to monitor and rank engagement.

7. ***Create 'OSS Parachute Team' that works with agencies that lack expertise and resources.***

   Our results in Chapters 3 and 4 found that some agencies have strong in-house OSS capabilities. Others, however, do not.

   One solution to this conundrum is for Code.Gov to create a specialist team that works with underperforming agencies – a form of in-government consultancy. Currently Code.Gov has one person to do this. A larger team could provide training and advice about how to establish OSS projects, and support community-building efforts once projects are underway.

   Benefits: Targets limited resources to agencies that most need support.

   Costs: Financial cost of hiring specialist team to work with agencies.

RECOMMENDATION

At a minimum, Code.Gov should provide training to federal acquisition employees, push for the Federal CIO Council to be convened, and consider launching an OSS parachute team.

These options strictly dominate the other options regardless of which policy objective Code.Gov decides to prioritize (Table 7).

If user engagement continues to be a core focus of the policy, then all of the above initiatives should be considered. This recommendation is somewhat endogenous: We compiled this list of potential initiatives after asking interviewees, "What practices could Code.Gov introduce – apart from being clear about its purpose and changing the volume requirement – to boost user

[104] Merelo-Guervos, Juan-Julian, Blancas, Israel, Arenas, Maribel G., Tricas, Fernando, Vacas, José Antonio, and Rico, Nuria, GitHub rankings and its impact on the local free software development community, *The Winnower* 2:e142251.14740, 2015.

engagement?" All of them therefore score +1 in the evaluation system.

If Code.Gov decides that user engagement is not going to be part of Open Source Policy 2.0, then the evaluation in Table 7 is also helpful: It shows which of the above initiatives would support the government procurement and in-government innovation objectives.

IV. INVESTIGATE OPEN QUESTIONS RAISED BY THIS RESEARCH

This report has raised as many questions as it has answers for the future of Code.Gov.

Our analysis has provided new insights into the performance of the Pilot Program since its inception in August 2016. However, it was restricted by factors outlined in Chapter 2.

As a result, there are outstanding questions that should be answered prior to the design of Open Source 2.0. These are:

1. HOW DOES ENGAGEMENT VARY BY LICENSING TYPE?

The type of license a repository uses can affect the community's willingness to engage with it.

The logic is that if a member of the open source community believes that there is a chance that their work will later be used for another party's commercial gain, they are less likely to engage with it in the first place.

The scraping tool that we developed for this analysis was unable to capture the licensing status of repositories. We were therefore unable to conduct this analysis. However, it may be an important consideration in the redesign of the policy. If the empirics confirm that top-performing repositories overwhelmingly use a specific licensing format, then Code.Gov may consider mandating in the future that all federal source code uses that format. If the empirics do not support this hypothesis, then the analysis may help debunk a common myth that has been repeatedly raised in our conversations for this project.

2. HOW DOES REUSABILITY CORRELATE WITH ENGAGEMENT?

Multiple sources told us that a repository's reusability drives engagement with repositories.

We were unable to test this hypothesis: Because neither of us are expert coders, we could not assess the modularity and reuse potential of repositories.

Code.gov should conduct an empirical analysis of this hypothesis. Our quantitative findings provide the roadmap for how to do this. A random sample of repositories from different quintiles (for example) could be selected. An expert coder could then parse these repositories to determine how complete, self-contained and usable they are.

Our analysis assumes that there would be a statistically significant difference in the number of repositories meeting the standard of completion, self-containedness and usability across quintiles – that is, that top-performing repositories would have a larger share of complete, self-contained and usable code than lower-performing repositories.

3. WHO IS ENGAGING WITH FEDERAL REPOSITORIES?

We have not been able to analyze the community of individuals engaging with federal source code.

The scraping tool that we developed with Code.Gov could not collect data on the users of federal repositories. This was due to restrictions with GitHub's API and privacy concerns.

Finding out who engages with federal source code matters. If most users are expert coders affiliated with the federal government (for example, technology consultants), then Code.Gov would be able to determine that the primary benefit of the Pilot Program is that it has facilitated easier improvement in the quality of federal code.

If, however, the majority are amateur coders or other organizations seeking to repurpose code for their own use, then it might be acceptable for projects to be developed in private before being released as open source upon completion.

Our interviews revealed that there is already work being done on this question. The Laboratory for Innovation Science at Harvard (LISH), for example, is working in conjunction with the Linux Foundation to conduct a census of all open source projects and their usage. We have had a preliminary conversation with the Linux Foundation about this work; there may be opportunities for Code.Gov to cooperate with LISH in order to better understand the user base of federal repositories.

4. WHAT FINANCIAL SAVINGS HAS THE PILOT PROGRAM GENERATED?

We have not focused on the commercial aspects of the Pilot Program.

However, the financial case for OSS was one reason that the Obama Administration launched the Pilot Program.

For example, Objective 3 – harnessing public contributions to government source code – might have indirectly resulted in savings, since the developer community was used to make improvements to code instead of contractors.

V. POLICY ARCHETYPES

We have made the above recommendations based on the assumption that user engagement will continue to be a part of Open Source Policy 2.0. In this section, we provide archetypes to illustrate how the recommendations might change if Code.Gov opts to prioritize different objectives.

The purpose is to illustrate the downstream implications of choosing a specific policy objective. We believe that there is an underappreciation of the impact of that decision on subsequent policy design. By providing these archetypes, we hope to demonstrate the extent to which the initial choice of objective will impact the shape of Open Source Policy 2.0. While each archetype includes 'Default to Open,' the other policy decisions change depending on the purpose.

1. GOVERNMENT PROCUREMENT

Purpose: To create cost savings and improve procurement practices by avoiding "duplicative custom software purchases" amongst agencies and vendor lock-in.

Decision 1: Default to open. If politically unfeasible, focus on maximizing how much source code agencies release.

Decision 2: Include provisions for training government acquisition, convening Federal CIO Council, and a "OSS Parachute Team."

Decision 3: Prioritize analysis of financial savings from Federal Source Code Policy.

2. IN-GOVERNMENT INNOVATION

Purpose: To fuel innovation within government software engineering by promoting "collaboration across Federal agencies" on projects.

Decision 1: Default to open. If politically unfeasible, focus on maximizing how much source code agencies release.

Decision 2: Include provisions for training government acquisition, convening Federal CIO Council, and a "OSS Parachute Team."

Decision 3: Prioritize analysis of how licenses impact contributions from developer community.

3. USER CONTRIBUTIONS

Purpose: To encourage qualitative improvement of federal source code by the public so it is "reliable and effective in furthering our national objectives."

Decision 1: Default to open. If politically unfeasible, focus on maximizing how much source code agencies release.

Decision 2: Include provisions for community-building training with all agencies, advocate for hiring full-time community managers, and publish regularly updated agency rankings on user engagement in open source.

Decision 3: Prioritize analysis of license type and of user demographics.

4. THIRD-PARTY REUSE

Purpose: To honor public ownership of the code and promote third party reuse, by providing the public with "the People's Code" that their taxpayer dollars fund.

Decision 1: Default to open. If politically unfeasible, focus on maximizing how much source code agencies release.

Decision 2: Include provisions for community building trainings with all agencies, advocate for hiring full-time community managers, and publish regularly updated agency rankings on user engagement in open source.

Decision 3: Prioritize analysis of user demographics.

APPENDIX

A. RELEVANT SECTION FROM FEDERAL SOURCE CODE POLICY

Section 5 (“Open Source Software”) of the Federal Source Code Policy, which outlines the Pilot Program

5. Open Source Software

5.1 Pilot Program: Publication of Custom-Developed Code as OSS

Each agency shall release as OSS at least 20 percent of its new custom-developed code29 each year for the term of the pilot program. As discussed above, agencies must obtain sufficient rights to custom-developed code to fulfill the open source release objectives of this policy’s pilot program.

When deciding which custom-developed code projects to release, each agency should prioritize the release of custom-developed code that it considers potentially useful to the broader community. Agencies should calculate the percentage of source code released using a consistent measure—such as real or estimated lines of code, number of self-contained modules, or cost—that meets the intended objectives of this requirement. Additional information regarding how best to measure source code will be provided on Code.gov.

Although the minimum requirement for OSS release is 20 percent of custom-developed code, agencies are strongly encouraged to release as much custom-developed code as possible to further the Federal Government’s commitment to transparency, participation, and collaboration.

OMB expects all agencies to satisfy the requirements of this pilot program without exception. Agencies should—as part of their selection of custom-developed code to be released as OSS—refrain from selecting code that would fall under the exceptions outlined in Section 6 of this policy. In the event that an agency’s CIO believes that the agency cannot satisfy the 20 percent requirement of the OSS pilot program (e.g., because releasing code as OSS would create an identifiable risk to the detriment of national security), the CIO should consult with OMB.

Unless extended or supplanted by OMB through the issuance of further policy, the pilot program under this sub-section will expire three years (36 months) after the publication date of this policy; however, the rest of the Federal Source Code Policy will remain in effect. No later than two years after the publication date of this policy, OMB shall evaluate pilot results and consider whether to allow the pilot program to expire or to issue a subsequent policy to continue, modify, or increase the minimum requirements of the pilot program.

Within 120 days of the publication date of this policy, OMB shall develop metrics to assess the impact of the pilot program. Additional information on these topics will be available on Code.gov.

5.2 Participation in the Open Source Community

When agencies release custom-developed source code as OSS to the public, they should develop and release the code in a manner that (1) fosters communities around shared challenges, (2) improves the ability of the OSS community to provide feedback on, and make contributions to, the source code, and (3) encourages Federal employees and contractors to contribute back to the broader OSS community by making contributions to existing OSS projects. In furtherance of this strategy, agencies should comply with the following principles:

Leverage Existing Communities: Whenever possible, teams releasing custom-developed code to the public as OSS should appropriately engage and coordinate with existing communities relevant to the project. Government agencies should only develop their own communities when existing communities do not satisfy their needs.

Engage in Open Development: Software that is custom-developed for or by agencies should, to the extent possible and appropriate, be developed using open development practices. These practices provide an environment in which OSS can flourish and be repurposed. This principle, as well as the one below for releasing source code, include distributing a minimum viable product as OSS; engaging the public before official release;30 and drawing upon the public's knowledge to make improvements to the project.

Adopt a Regular Release Schedule: In instances where software cannot be developed using open development practices, but is otherwise appropriate for release to the public, agencies should establish an incremental release schedule to make the source code and associated documentation available for public use.

Engage with the Community: Similar to the requirement in the Administration's Open Data Policy, agencies should create a process to engage in two-way communication with users and contributors to solicit help in prioritizing the release of source code and feedback on the agencies' engagement with the community.

Consider Code Contributions: One of the potential benefits of OSS lies within the communities that grow around OSS projects, whereby any party can contribute new code, modify existing code, or make other suggestions to improve the software throughout the software development lifecycle. Communities help monitor changes to code, track potential errors and flaws in code, and other related activities. These kinds of contributions should be anticipated and, where appropriate, considered for integration into custom-developed Government software or associated materials.

Documentation: It is important to provide OSS users and contributors with adequate documentation of source code in an effort to facilitate use and adoption. Agencies must ensure that their repositories include enough information to allow reuse and participation by third parties. In participating in community-maintained repositories, agencies should follow community documentation standards. At a minimum, OSS repositories maintained by agencies must include the following information:

- Status of software (e.g., prototype, alpha, beta, release, etc.);
- Intended purpose of software;
- Expected engagement level (i.e., how frequently the community can expect agency activity);
- License details; and
- Any other relevant technical details on how to build, make, install, or use the software, including dependencies (if applicable).

B. RAW CODE USED TO GENERATE DATA

We worked with the Code.Gov team – particularly Joe Castle and Froilan Irizarry – to develop custom code that pulls data about federal repositories from GitHub. This code is open source and available at the links listed below.

i. Repository-level data

Available at https://github.com/froi/us-federal-gov-github-orgs-stats/blob/master/graphql/repo_data.gql

```
query($search_query: String!, $size: Int!, $cursor: String) {
  search(first:$size, query:$search_query, type: REPOSITORY, after:$cursor){
    repositoryCount
    edges {
      node {
        __typename
        ... on Repository {
          name
          owner {
            login
          }
          issues {
            totalCount
          }
          forks {
            totalCount
          }
          stargazers {
            totalCount
          }
          watchers {
            totalCount
          }
          forkCount
          nameWithOwner
          createdAt
          isPrivate
        }
      }
    }
    pageInfo {
      endCursor
      hasNextPage
    }
  }
}
```

ii. Issue and pull request data

Available at https://github.com/froi/us-federal-gov-github-orgs-stats/blob/master/graphql/issues_data.gql

```
query($search_query: String!, $size: Int!, $cursor: String) {
  search(first:$size, query:$search_query, type: ISSUE, after:$cursor){
    issueCount
    edges {
      node {
        __typename
        ... on Issue {
          title
          createdAt
          lastEditedAt
          state
          updatedAt
          repository {
            name
            owner {
              login
            }
          }
        }
        ... on PullRequest {
          title
          createdAt
          lastEditedAt
          state
          updatedAt
          repository {
            name
            owner {
              login
            }
          }
        }
      }
    }
    pageInfo {
      endCursor
      hasNextPage
    }
  }
}
```

iii. Star and fork data

Available at https://github.com/froi/us-federal-gov-github-orgs-stats/blob/master/graphql/stars_forks_data.gql

```
query($org: String!, $repoName: String!, $starsCursor: String, $forksCursor: String, $size: Int) {
  repository(owner: $org, name: $repoName) {
    stargazers(first: $size, after: $starsCursor, orderBy: {field: STARRED_AT, direction: ASC}) {
      edges {
        starredAt
      }
      pageInfo {
        endCursor
        hasNextPage
      }
    }
    forks(first: $size, after: $forksCursor, orderBy: {field: CREATED_AT, direction: ASC}) {
      edges {
        node {
          createdAt
        }
      }
      pageInfo {
        endCursor
        hasNextPage
      }
    }
  }
  rateLimit {
    limit
    remaining
    resetAt
  }
}
```

C. RANKINGS OF AGENCIES BY ENGAGEMENT

Agencies, sorted by total number of engagements

Rank	Agency	No. engagements	No. repos	Average no. engagements per repository
1	NASA	56,449	331	0.00586
2	Department of Defense	37,071	360	0.00971
3	General Services Administration	22,695	1,098	0.04838
4	Department of the Interior	16,482	781	0.04739
5	Department of Health and Human Services	11,222	274	0.02442
6	Department of Commerce	10,090	254	0.02517
7	Department of Energy	4,971	217	0.04365
8	Department of State	3,102	150	0.04836
9	Department of Veterans Affairs	3,085	146	0.04733
10	Environmental Protection Agency	2,105	105	0.04988
11	Department of Labor	1,471	63	0.04283
12	Department of the Treasury	1,410	19	0.01348
13	Department of Justice	1,404	31	0.02208
14	Department of Homeland Security	1,205	10	0.0083
15	US Department of Agriculture	995	31	0.03116
16	Office of Personnel Management	930	4	0.0043
17	Department of Transportation	555	26	0.04685
18	Small Business Administration	500	11	0.022
19	Department of Education	123	4	0.03252
20	United States Agency for International Development	91	5	0.05495
21	Social Security Administration	77	2	0.02597
22	National Science Foundation	5	2	0.4

Agencies, sorted by average engagements per repository

Rank	Agency	No. engagements	No. repos	Average no. engagements per repository
1	National Science Foundation	5	2	0.4
2	United States Agency for International Development	91	5	0.055
3	Environmental Protection Agency	2,105	105	0.050
4	General Services Administration	22,695	1,098	0.048
5	Department of State	3,102	150	0.048
6	Department of the Interior	16,482	781	0.047
7	Department of Veterans Affairs	3,085	146	0.047
8	Department of Transportation	555	26	0.047
9	Department of Energy	4,971	217	0.044
10	Department of Labor	1,471	63	0.043
11	Department of Education	123	4	0.033
12	US Department of Agriculture	995	31	0.031
13	Social Security Administration	77	2	0.026
14	Department of Commerce	10,090	254	0.025
15	Department of Health and Human Services	11,222	274	0.024
16	Department of Justice	1,404	31	0.022
17	Small Business Administration	500	11	0.022
18	Department of the Treasury	1,410	19	0.013
19	Department of Defense	37,071	360	0.010
20	Department of Homeland Security	1,205	10	0.010
21	NASA	56,449	331	0.006
22	Office of Personnel Management	930	4	0.004

D. RANKINGS OF AGENCY RESPONSIVENESS

Table: Number and status of Issues flagged

Agency	**Closed**	**Open**	**Total**	**% remaining open**
Department of the Treasury	7	12	19	63%
USAID	15	18	33	55%
Department of Veterans Affairs	203	243	446	54%
General Services Administration	1,074	1,015	2,089	49%
Small Business Administration	12	11	23	48%
Environmental Protection Agency	432	390	822	47%
Department of Justice	357	270	627	43%
Department of Education	46	29	75	39%
NASA	1,815	1,057	2,872	37%
Department of the Interior	2,718	1,434	4,152	35%
Department of Homeland Security	518	270	788	34%
Department of Labor	393	180	573	31%
Department of Health and Human Services	738	332	1,070	31%
Department of Commerce	1,356	574	1,930	30%
Department of State	941	391	1,332	29%
Department of Energy	831	320	1,151	28%
Department of Defense	2,531	822	3,353	25%
US Department of Agriculture	580	152	732	21%
National Science Foundation	177	43	220	20%
Social Services Administration	34	8	42	19%
Department of Transportation	107	23	130	18%
Office of Personnel Management	884	92	976	9%

Agency	Closed	Merged	Open	% Acted on	Total
Office of Personnel Management	5	19	-	100%	24
USAID	-	7	-	100%	7
Small Business Administration	33	485	2	100%	520
Department of Justice	83	464	6	99%	553
Department of Transportation	55	249	4	99%	308
Department of Labor	24	197	3	99%	224
Department of the Treasury	133	830	20	98%	983
Department of State	78	1,013	23	98%	1,114
Department of Commerce	221	2,278	55	98%	2,554
Department of Health and Human Services	194	2,782	68	98%	3,044
Environmental Protection Agency	23	230	7	97%	260
Department of the Interior	281	4,238	133	97%	4,652
Department of Defense	364	3,514	140	97%	4,018
NASA	325	2,367	121	96%	2,813
National Science Foundation	5	57	3	95%	65
Department of Homeland Security	23	180	10	95%	213
Department of Veterans Affairs	102	1,369	83	95%	1,554
General Services Administration	304	2,719	224	93%	3,247
US Department of Agriculture	1	49	4	93%	54
Department of Energy	50	830	80	92%	960
Department of Education	-	8	1	89%	9

BIBLIOGRAPHY

WORKS CITED

"Affiliate Membership Qualifications and Criteria." *Affiliate Membership Qualifications and Criteria | Open Source Initiative*, opensource.org/AffiliateRequirements.

"Improving the Acquisition and Management of Common Information Technology: Software Licensing". Office of Mgmt. & Budget, Exec. Office of the President, June 2, 2016. Available at: https://obamawhitehouse.archives.gov/sites/default/files/omb/memoranda/2016/m-16-12_1.pdf.

"Measuring Source Code", *Code.Gov*, https://code.gov/about/open-source/measuring-code.'

"Data.ca.gov." *Data.ca.gov*, data.ca.gov/.

"Dwyl/Repo-Badges." *GitHub*, 29 Nov. 2018, github.com/dwyl/repo-badges.

"Making Source Code Open and Reusable." GOV.UK, www.gov.uk/service-manual/technology/making-source-code-open-and-reusable.

"Metrics With Greater Utility: The Community Manager Use Case." CHAOSS, 25 Feb. 2019, chaoss.community/news/2018/11/16/metrics-with-greater-utility-the-community-manager-use-case/.

"Open Data | Open Data NY." *State of New York*, data.ny.gov/.

"Project Interoperability." Project Interoperability, project-interoperability.github.io/.

"Proprietary Software vs. Open Source - The Hidden Costs." *Trellon*, trellon.com/content/blog/proprietary-software-vs-open-source-hidden-costs.

"The People's Code." *National Archives and Records Administration*, National Archives and Records Administration, obamawhitehouse.archives.gov/blog/2016/08/08/peoples-code.

"Three-Step Software Solutions Analysis." *Three-Step Software Solutions Analysis*, policy.cio.gov/source-code/three-step-software-solutions-analysis/.

Balter, Ben. *Towards a More Agile Government*. 29 Nov. 2011, ben.balter.com/2011/11/29/towards-a-more-agile-government/#fn:2.

Fogel, Karl, *Producing Open Source Software: How to Run a Successful Free Software Project*, Version 2.3098, Available online at http://producingoss.com.

Goldstein, Phil, "Federal Agencies Will Be Required to More Accurately Track Software Licenses." *Technology Solutions That Drive Government*, 24 Aug. 2016, fedtechmagazine.com/article/2016/08/federal-agencies-will-be-required-more-accurately-track-software-licenses.

Government Digital Service. "8. Make All New Source Code Open." *GOV.UK*, GOV.UK, 29 June 2016, www.gov.uk/service-manual/service-standard/make-all-new-source-code-open.

Kesan, Jay P. & Shas, Rajiv C. Shah, Shaping Code, *Harvard Journal of Law & Technology*, Volume 18, Number 2 Spring 2005 pp 320-398

Kuldell, Heather. “It's Official: MEGABYTE Act Signed into Law.” *Nextgov.com*, Nextgov, 28 Nov. 2017, www.nextgov.com/cio-briefing/2016/08/its-official-megabyte-act-signed-law/130391/.

McDonald, Nora, and Goggins, Sean, Performance and Participation in Open Source Software on GitHub, CHI EA '13 CHI '13 *Extended Abstracts on Human Factors in Computing Systems* Pages 139-144

McKinsey and Company, *Big data: The next frontier for innovation, competition, and productivity* (2011). Available at https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Big%20data%20The%20next%20frontier%20for%20innovation/MGI_big_data_exec_summary.ashx

Merelo-Guervos, Juan-Julian, Blancas, Israel, Arenas, Maribel G., Tricas, Fernando, Vacas, José Antonio, and Rico, Nuria, GitHub rankings and its impact on the local free software development community, *The Winnower* 2:e142251.14740, 2015.

Mill, Eric, et al. “Digital Service Delivery | How We Built Analytics.usa.gov.” *18F*, 19 Mar. 2015, 18f.gsa.gov/2015/03/19/how-we-built-analytics-usa-gov/.

Nagle, Frank, Learning by Contributing: Gaining Competitive Advantage Through Contribution to Crowdsourced Public Goods, *Organization Science*, *Organization Science*, 2018, Vol.29(4), pp. 569-587.

Ndenga, Malanga Kennedy, Jean, Mehat, Ganchev, Ivaylo, and Franklin, Wabwoba Assessing Quality of Open Source Software Based on Community Metrics, *International Journal of Software Engineering and Its Applications*, (2015) 9:12, 337-348.

Roberts, Jeffrey A., Hann, Il-Horn, and Slaughter, Sandra A., Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects, *Management Science* Vol. 52, No. 7, Open Source Software (Jul., 2006), pp. 984-999.

Rubens, Paul. “Open Source Code Contains Fewer Defects, But There's a Catch.” *CIO*, CIO, 18 Nov. 2014, www.cio.com/article/2847880/open-source-code-contains-fewer-defects-but-theres-a-catch.html.

Scott, Tony, U.S. Chief Information Officer. “The People's Code.” *National Archives and Records Administration*, National Archives and Records Administration, 2016, obamawhitehouse.archives.gov/blog/2016/08/08/peoples-code.

Senz, Kristen, The Hidden Benefit of Giving Back to Open Source Software, *HBS Working Knowledge*, September 5 2018, https://hbswk.hbs.edu/item/the-hidden-benefit-of-giving-back-to-open-source-software.

Shipman, Anna. “Don't Be Afraid to Code in the Open: Here's How to Do It Securely.” *Technology in Government*, gdstechnology.blog.gov.uk/2017/09/27/dont-be-afraid-to-code-in-the-open-heres-how-to-do-it-securely/.

Shipman, Anna. “The Benefits of Coding in the Open.” *Government Digital Service*, gds.blog.gov.uk/2017/09/04/the-benefits-of-coding-in-the-open/.

Sushchenia, Iryna and Grönlund, Åke, Organizational measures to stimulate user engagement with open data, *Transforming Government: People, Process and Policy* Vol. 9 No. 2, 2015 pp. 181-206

Tomassetti, Federico , and Torchiano, Marco, An Empirical Assessment of Polyglot-ism in GitHub - *EASE ’14*, May 13 - 14 2014.

U.S. Department of Defense. “Contracts for January 11, 2019.” *U.S. Department of Defense*, dod.defense.gov/News/Contracts/Contract-View/Article/1730557//.

Zorz, Zeljka. “The Percentage of Open Source Code in Proprietary Apps Is Rising.” *Help Net Security*, 22 May 2018, www.helpnetsecurity.com/2018/05/22/open-source-code-security-risk/.

Zvenyach, V. David. “The Trouble with the Federal Source Code Policy, and What to Do about It: Part One.” *Medium*, 9 Oct. 2018, medium.com/@vdavez/the-trouble-with-the-federal-source-code-policy-and-what-to-do-about-it-part-one-f1f26d0232ab.

Zvenyach, V. David. “The Trouble with the Federal Source Code Policy, and What to Do about It: Part One.” *Medium*, 9 Oct. 2018, medium.com/@vdavez/the-trouble-with-the-federal-source-code-policy-and-what-to-do-about-it-part-one-f1f26d0232ab.

WORKS CONSULTED

"Apache Way." *Apache Foundation,* https://www.apache.org/foundation/how-it-works.html.

"Open Source Guides", *GitHub, https://opensource.guide/best-practices/*.

"Open Source", *Google, https://opensource.google.com/docs/*.

"Best Practices for Open Source in Government (Using GitHub)." *DigitalGov*, 6 Nov. 2013, digital.gov/2013/11/06/github-for-government/.

"Best Practices", *OpenOffice, https://www.openoffice.org/docs/bestpractices.html.en*.

"Curriculum." *OpenChain*, www.openchainproject.org/curriculum.

"Open Source for America." *Open Source for America*, opensourceforamerica.org/.

"Tools." *Codice*, codice.org/tools.html.

Begel, Andrew, Bosch, Jan and Storey, Margaret-Anne, Social Networking Meets Software Development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder, *IEEE Software,* Volume: 30 , Issue: 1, Jan.-Feb. 2013.

Biazzini, Marco, and Baudry, Benoit, "May the fork be with you": novel metrics to analyze collaboration on GitHub, WETSoM 2014 Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics Pages 37-43, Available at: https://dl.acm.org/citation.cfm?id=2593875

Blair, Anthony L., Best Practices: Open Source Curriculum, *The Journal of Continuing Higher Education*, 54:1, 28-33, 2006.

English, R. and Schweik, C.M. 2007. "Identifying Success and Abandonment of Free/Libre and Open Source (FLOSS) Commons: A Preliminary Classification of Sourceforge.net projects." *Upgrade: The European Journal for the Informatics Professional*. Vol. VIII, Issue no. 6 (December). Available at http://www.upgrade-cepis.com/issues/2007/6/upg8-6English_Schweik_v2.pdf

Feller, Joe, Fitzgerald, Brian, Hissam, Scott and Lakhani, Karim R., eds. *Perspectives on Free and Open Source Software. Cambridge: MIT Press, 2005.*

Gustafson, Britta, and Will Slack. "Digital Service Delivery | Facts about Publishing Open Source Code in Government." *18F*, 8 Aug. 2016, 18f.gsa.gov/2016/08/08/facts-about-publishing-open-source-code-in-government/.

Harhoff, Dietmar and Lakhani, Karim R. , eds. Revolutionizing Innovation: Users, Communities, and Open Innovation. Cambridge, MA: MIT Press, 2016.

Jarczyk O., Gruszka B., Jaroszewicz S., Bukowski L., Wierzbicki A. (2014) GitHub Projects. Quality Analysis of Open-Source Software. In: Aiello L.M., McFarland D. (eds) *Social Informatics. SocInfo* 2014. Lecture Notes in Computer Science, vol 8851, 2014.

Weber, Steven, *The Success of Open Source*, Harvard University Press, 2004

West, Joel, and Lakhani, Karim R. . "Getting Clear About Communities in Open Innovation." *Industry and Innovation* 15, no. 2 (April 2008).

IMAGE CREDITS

All images have with non-commercial reuse licenses. They are available at:

- *Chapter 1* - https://www.flickr.com/photos/hackny/6890140478
- *Chapter 2* - https://commons.wikimedia.org/wiki/File:Wikimedia_Hackathon_2013,_Amsterdam_-_Flickr_-_Sebastiaan_ter_Burg_(28).jpg
- *Chapter 3* - https://www.pexels.com/photo/two-women-looking-at-the-code-at-laptop-1181263/
- *Chapter 4* - https://media.defense.gov/2015/Oct/14/2001299878/600/400/0/140610-Z-PA893-125.JPG
- *Chapter 5* - https://pxhere.com/en/photo/7742