

NYC Department of Health and Mental Hygiene’s Environmental Health Data Portal: An Open Path Forward

9 July 2020

Introduction	3
1 Executive Summary	4
2 Open Source As A Strategic Tool	5
2.1 Ecosystem Map	7
2.2 Vision Setting	8
2.3 Choosing Open Source Goals	11
2.4 Identifying Archetypes For Best Practices	18
3 Embracing Open	21
3.1 Public Communications	24
3.2 Open Source Infrastructure	25
4 Open Source Municipal Procurement	29
4.1 Modular Contracting	30
4.2 Intellectual Property Contract Terms	33
4.3 Open Source Quality Assurance	35
4.4 Staffing	37
4.5 Budgeting	39
4.6 Open Source Solicitation	42

5	Sketching A New Portal	44
5.1	Audiences	45
5.2	Architecture	47
5.3	User Interface Concerns	53
5.4	Editing and Admin Workflow	54
6	Thanks	57
7	Acronyms	58
	Appendix A: Open Source Analysis Checklist	58
	A.1 How Does This Project Thrive As Open Source?	59
	A.2 Implementing An Open Source Strategy	61
	Appendix B: OSQA Example SOW	63
	Appendix C: Ecosystem Maps	68
	Appendix D: Ecosystem Mapping Worksheet	72
	Appendix E: Open Source Goal Setting Worksheet	73

Introduction

New York City’s Department of Health and Mental Hygiene (DOHMH) operates a web portal that provides data about factors affecting environmental health in the City.¹ That portal serves citizens, journalists, and researchers both within the City and beyond. It has grown to provide both data and narrative context that helps people understand how the numbers impact human health.

Although the portal is well-suited to its purpose, the wider technology industry has in recent years undergone improvements in data platforms and in software development processes. Those advances present opportunities for DOHMH to upgrade the portal to improve DOHMH’s ability to serve its audiences.

This document is the culmination of an effort by Open Tech Strategies (OTS) to aid DOHMH in seizing those opportunities. OTS examined the portal and conducted a series of interviews with current and former City staff. This final report contains recommendations for a new technical structure and an agile,² open source development process, the combination of which would allow DOHMH to incrementally improve the portal using a flexible staffing arrangement sensitive to the needs of a city agency whose central mission is not focused on technology development.

By the end of this document, readers should have a basic understanding of:

- How the environmental data portal might respond to an open source approach
- A high-level architecture plan for a new portal
- A plan to integrate compatibly with the current portal
- How an NYC agency can procure an open source portal

More generally, both DOHMH and OTS intend for this document to provide guidance to agencies in approaching open source. The process described below is specific to DOHMH’s needs, but similar analysis might apply to any other City software project. We offer this document as a starting point for any municipal agency’s use as it makes plans regarding its technology needs.

¹<http://a816-dohbep.nyc.gov/IndicatorPublic/>

²Throughout this document, we use lower-case “agile” in an informal sense, to refer to a flexible and iterative style of software development that has much in common with upper-case “Agile” as formally described in the “Manifesto for Agile Software Development” (<http://agilemanifesto.org/>) but that does not necessarily adhere to every principle laid out in that manifesto. See https://en.wikipedia.org/wiki/Agile_software_development for more on the history of “agile”, “Agile”, and related methodologies.

1 Executive Summary

This document offers a number of specific recommendations,³ and at a high level they boil down to these three:

- **Implement an open source strategy.** An explicitly open source strategy will serve DOHMH very well in further developing the environmental health portal. We explain what this means in Sections 2 and 3.
- **Make procurement and contracting adjustments to better support open source, agile development.** Open source strategy goes hand-in-hand with iterative, responsive development techniques, which in turn require contracts to be structured to support this kind of development. Section 4 discusses this in detail.
- **Use technologies and development processes that will reap the greatest benefit from an open source strategy.** Section 5 covers these recommendations, in particular discussing technology platform choices in some depth.

Below we summarize some of the most important of the key recommendations. Although this is far from a complete list of every piece of advice presented in this document, it attempts to be representative: reading the list below should accurately characterize our overall analysis while not overwhelming with details.

- Create solicitation procedures that attract vendors who understand open source natively, and structure contracts so that normal amounts of iteration and flexibility are predicted and built in, instead of being treated as exceptions that require contract amendments. See Section 4.6.
- Choose a limited set of specific open source goals and explicitly prioritize them over other possible open source goals, in order to guide decision-making and investments. Section 2.3 explains the three goals we think would be most appropriate for DOHMH: *improving internal collaboration*, *breaking vendor lock-in*, and *creating a framework for partner collaboration*.
- Use ecosystem mapping to improve DOHMH's ability to identify and interact with partners (paid and otherwise) and usage communities. See Section 2.1 for more.
- Implement Open Source Quality Assurance (OSQA) as an organizing principle of the development and communications process, both internally and when interacting with collaborating vendors. Section 4.3 describes this in detail.
- Ensure that the ownership clauses (copyright, patents, trademarks) of all development contracts ensure DOHMH full non-exclusive rights to use, modify, and redistribute the work product as open source software. See Section 4.2.

³Many of them appear in “Key Recommendation” boxes that are designed to stand out visually.

- Adopt open source technical collaboration practices, such as using a public version control repository, bug tracker, etc. See section 3.2 for details.
- Use real-world usage scenarios to create personas and user stories for key audience segments, to help ensure that design and content serve all constituencies well. Section 5.1 discusses this more. Furthermore, obtain more information about usage through analytics, as described in 5.4.1.
- Improve the internal data production workflow, by improving both tooling and training. See Section 5.4.
- Prioritize requirements, starting with improvements to content, and then moving on to improved support for workflow around editing and approvals. Section 5 contains further discussion of this topic.
- Don't treat Maintenance & Operations (M&O) monolithically. Operations may be handled by an entirely different team from the development team, while maintenance work should be carried out, as much as possible, as part of the continuous cycle of regular development that results in ongoing, iterative improvements to the system. Section 4.5.1 explains this in more depth.
- Consider using a Django-based back end and a Bootstrap-based front end (with React components brought in as necessary) as default technology choices for the next generation of the portal. Section 5.2.1 explains why.

DOHMH is well-positioned to embrace these opportunities. Its current technical expertise is unusually strong compared to many municipal agencies OTS is familiar with. DOHMH is, in most areas that matter, up-to-date with modern industry practice. While the agency might make wider use of that expertise (e.g., deepen its investment in user research), it is clear that work is already proceeding carefully and according to high standards.

One of the challenges facing DOHMH is turning its data into practical impact. In one sense, DOHMH does its job just by providing public information about environmental health factors. From a mission perspective, though, success requires that data to have real-world effects: it should contribute to changes in decisions and policy that affect human welfare here in New York. Increasing that impact is the justification for improving the portal.

Inherent in the vision of the data portal is the notion that people will find their own stories in the data. Even as DOHMH always seeks to tell those stories better, it is just as important to help visitors explore the data that leads them to their own insights.

2 Open Source As A Strategic Tool

Open source software is now a vital part of the technology landscape. Virtually every modern computer server, laptop, and cellphone depends on a large amount of open source

software to operate, and Free and Open Source Software (FOSS)⁴ is widespread in digital electronics such as wireless routers, printers, cameras, etc as well.

Open source software is software code that is published under open copyright: anybody can run, copy, modify, and distribute it. The typical open source licenses in common use differ in a few details, but they all guarantee those basic freedoms, and thus they create a common pool of software code that everyone can use and improve on. This approach stands in contrast to “proprietary” software, which comes encumbered with various monopoly restrictions and generally does not include permission for reuse, deep customization, or sharing.

While FOSS is very widely used, merely *using* it is only part of the story. FOSS also affects the structure and practices of the organizations that interact with it, because FOSS development tends to strongly reward collaborative investment. Even when the collaboration happens “asynchronously” — i.e., one organization picking up later where another left off earlier — over the long term all the parties are still rewarded by the collaboration.

Many companies already use FOSS as their primary model for working with other firms on joint projects. FOSS routinely enables teams of loosely related developers separated by various types of boundaries to produce industry-leading software. Sometimes, those boundaries might be between teams and functional units inside an agency. For other projects, development efforts might be spread across a city government and multiple jurisdictions, or involve work completed over long periods of time, spanning multiple administrations and proceeding in fits and starts. In such situations, many companies, governments, and NGOs have turned to open source principles to spur innovation and cooperation.

Because municipal agencies often deal with more organizational and jurisdictional boundaries than are found in the private sector, municipalities have, in a sense, a greater opportunity to realize the benefits of FOSS collaboration. DOHMH in particular is well-positioned to take advantage of this dynamic. It has built software with both in-house developers and vendors (the Environmental Health Data Portal, and the Rat Information Portal before that), and it has in-house expertise in agile development methodology, user experience design, and cutting-edge Drupal development. All of these involve working with open source processes and artifacts.

What remains is for DOHMH to fully realize the innovation and quick-forming cooperative relationships that arise when open, non-monopolistic terms are the basis of collaboration. Built on top of FOSS licensing is a set of norms and collaboration practices that have evolved from decades of distributed development. There are shared technical and cultural conventions about how to communicate, review and approve proposed changes, write documentation, manage customizations and versions, etc.

Together they form a set of open source best practices that allow diverse collaborators —

⁴“Open source” software is also sometimes called by the older term “free software.” For most purposes, these terms are interchangeable. The expression “FOSS” is often used, as we use it here, for inclusivity and to avoid any ambiguity.

partners, vendors, users, and even competitors — to make competitive, world-class software. These practices create the trust and transparency that encourage others to invest in one’s technology, and they facilitate a continuous flow of information about what different stakeholders actually need from the software.

The question for DOHMH is: why apply a structure that enables and benefits from widespread collaboration to a tool that will, at least initially, be deployed by just one agency? We address this more deeply in Section 2.3, but for now it is enough to understand that even internal projects are collaborations by contributors separated by various types of boundaries.⁵

Achieving beneficial open source dynamics requires effort. It does not come just from applying an open source license and publishing a codebase. Open source is a *collaborative process*, which is to say that investing in open source requires investing in that collaboration. These investments should be made with tangible goals in mind, and the specific objectives chosen (see Section 2.3 “Choosing Open Source Goals”) will affect strategic decision-making.

One thing open source is not is a business model or a sustainability model. It is a collaboration model backed by norms and standardized legal infrastructure. Open source can support many sustainability models (including many of the same models proprietary software uses), but it is not an entire strategy in and of itself. When planning, it is crucial to identify concrete goals and the mechanisms by which open source helps to achieve those goals.

With that backdrop in mind, OTS applied a set of analytic tools to the portal effort to clarify goals and potential collaboration opportunities. We present that work in the remainder of this section, both to support analysis of the portal and as a step to enabling DOHMH to perform such analysis on future projects.

2.1 Ecosystem Map

Because open source is a collaboration model, it is impossible to fashion an open source approach without understanding the constellation of contributors, stakeholders, and related projects that will surround the effort. One of the first steps of open source planning is to generate a high-level view of a project’s potential or actual ecosystem. We do this both to help understand the environment and to establish a common view among everybody involved in project planning.

Start with a map of current and potential actors. This can be done as an individual assignment, but is usually much more effective as a group exercise done with a whiteboard. List service providers and group them by the type of service they offer. Identify potential collaborators, and mark efforts that compete for attention or resources. Identify any substitutes for your project, regardless of whether they are open or proprietary. Place actors with large, current impact closer to the center of the map and future, potential

⁵e.g., in-house vs. vendors, designer vs. coder, project manager vs. developer, etc.

participants further away. The open source project belongs at the center of the map, and DOHMH itself might be close in or further out, depending on its current effective scale of involvement. For a project that DOHMH originates, DOHMH will be quite close to the center, at least initially. When done, note interesting relationships between various nodes on the map. Mark significant partners and add users in another color. This map is a picture of the world as it currently exists and how it might change in the near-term future. Be sure to save a snapshot of this map to see how it shifts over time.

The type of information recorded in a map and the way one visualizes the ecosystem will depend heavily on the type of strategic thinking the map supports. For example, in deciding where to invest resources, DOHMH might make a map that groups participants by agency or sector. In another scenario, DOHMH might contemplate involving states or reaching out to the Center for Disease Control (CDC) and other federal agencies. It might then group participants by jurisdiction or area of environmental health focus. Either way, let the strategic concerns dictate the map's focus, not the other way around.

One particular visualization that is useful in the governmental sector is a map of open source capability. Part of DOHMH's planning might focus on collaboration with other partners who are experienced or at least eager on open source matters. By planning cooperation with those most suited to work well in open source modes, DOHMH can maximize its chances of attracting the kind of participation it needs to succeed. Conversely, it does not make sense to expect adept open source cooperation from peers that are not sophisticated open source participants.

We did not conduct an ecosystem mapping session for this engagement. OTS cannot perform this exercise on its own, and assembling members of the DOHMH portal team for an on-site mapping session was beyond the scope of this engagement. We did, however, gather a wealth of information about the ecosystem during the research phase, and that informs the work presented in this report.



Key Recommendation: Conduct internal training and education on ecosystem mapping to increase open source capacity at DOHMH, and apply this new skill to the portal.

Please see Appendix C: “Ecosystem Maps” for more on ecosystem maps, including sample maps.

2.2 Vision Setting

At the highest level, a project relies on a clear vision to orient and prioritize all its activities. A good litmus test for any effort is the ability to ask participants to explain the big picture vision and how their individual tasks contribute to that vision. Organizations in which people can do this easily and confidently tend to operate more robustly than those that cannot meet this test.

OTS recommends explicitly specifying goals at multiple levels to help guide software development collaborations. It starts with high level vision that flows down to inform efforts throughout the agency. Generally, we find it useful to specify these at three levels:

- The top-line mission of your agency or group within DOHMH
- The overall goal of the project within your group
- The open source goals related to the specific project

Each one of these feeds the goals of the item above it. In addition to clearly defining organizational and project goals, planning also benefits from an explicit understanding of how each layer supports the one above. This exercise is designed to promote mission alignment and ensure that everybody's work is productive in ways that matter to the overall goals of the agency.

Thankfully, DOHMH's top-line mission is clear. DOHMH has a clearly defined mandate to "protect and promote the health of 8 million diverse New Yorkers."⁶ Similarly, each division or bureau within DOHMH has a clear mission. For the Bureau of Environmental Surveillance and Policy (BESP), that mission is "to ground DOHMH programs and policy in science and law to support a healthy and equitable natural, built and occupational environment for all New Yorkers." BESP's mission has several sub-goals:

- We promote environmental health through legal and data-driven program support.
- We use innovative methods to track environmental determinants of health with a focus on New Yorkers disproportionately impacted by environmental risk.
- We share data and research findings to encourage evidence-based decision-making.
- We partner with communities in participatory science and outreach to address stakeholder concerns and empower residents.

Specifying these high-level goals might seem pedantic, but it provides a framework for prioritizing the rest of our work. For example, it tells us that while the portal might count among its successes the improvement in health of people who are not New Yorkers, those achievements are, by the scope of the mission, in some ways less of a priority than protecting the health of the people here in New York.

The goals of individual projects within DOHMH will be more focused than the vision for the entire agency, and will encompass activities that are broader than the open source parts of the project. These goals will vary in scope, definition, and specificity. In some cases, project goals might be implicit or stated in an incomplete way. In those cases, this is an opportunity to bring increased clarity or at least specify how those goals are understood by the open source planning team. Where possible, it is best to establish project goals early in

⁶<https://www1.nyc.gov/site/doh/about/about-doh.page>

the planning process. Though these goals are not specifically about open source, they will provide the process with a sense of which open source goals are worth pursuing and why.

In the case of projects that already exist, of course, goals should be readily available. The portal is one such project. It has its own goals designed to serve the greater DOHMH mission. The portal provides “data on a variety of topics that show how the environment affects health.”⁷ It is also “part of the National Environmental Public Health Tracking Network, an effort led by the Centers for Disease Control and Prevention to share data and analyze trends in environmental public health across the nation.”⁸ This second goal of sharing data and tracking trends across the nation reaches beyond New York, but has a clear connection with promoting the health of people in New York. Putting all that together with DOHMH’s goals, we might say **the portal shares data on how environmental factors affect health, in order to improve the health of New Yorkers.**

We might approach this articulation of the project’s goals with some of BESP’s subgoals in mind. The portal data should be actionable in the hands of decision-makers. DOHMH might look for opportunities to partner with specific communities on collecting data or presenting it for their use. These communities might be neighborhoods or other groupings of citizens. Detailing how the portal addresses stakeholder concerns starts with identifying stakeholders and studying how they make use of the portal and its data. Fortunately, that’s already part of the team’s work. The subgoal of “empowering residents” places residents among the stakeholders and invites them into the decision-making conversation. With these subgoals in mind, we might refine our earlier statement: **The portal shares data on how environmental factors affect health, in order to improve the health of New Yorkers. We work with communities and stakeholders to support decision-making and empower residents.**

These early goal-setting steps should not be difficult. The goal statement does not need to be overly precise, and does not need to be a published mission statement. A project formed with sufficient clarity of vision should be able to state its aims and even to be explicit about where those aims start to lack clarity and focus. Recording these goals allows an effort to effectively communicate to potential partners⁹ (both within DOHMH and outside) the value of working together. It also allows a project to notice and effectively communicate when those goals *change*.

Note that it can sometimes be useful to state *non-goals* explicitly along with goals. This allows DOHMH to declare activities “out of scope” because they pertain to goals DOHMH does not care to pursue. That has the effect of clarifying the pursuit of the things it does care about.

So far, the initial goal-setting covered in this report has nothing to do with open source approaches to software. It hasn’t considered software at all. Rather, these are preliminary

⁷ *Id.*

⁸ <http://a816-dohbesp.nyc.gov/IndicatorPublic/LearnMore.aspx>

⁹ Note that it is common for a project’s goals to differ from those of specific participants. The value of FOSS methodology is that it allows productive cooperation to thrive despite disparate goals among collaborators.

steps. Explicitly specifying these broader goals enables DOHMH to undertake more focused goal-setting in the next step: setting open source goals.

2.3 Choosing Open Source Goals

Once DOHMH defines the vision for a project, it is possible to consider the specific benefits offered by open source approaches. There are a wide array of potential gains that open source projects create, and this analysis aims to pick the most important ones to focus on during strategic planning. This stage is where a project might expect to spend more time planning, and also where goals might be subject to change in response to shifts in the operating environment. The key question to answer is “What are the effects DOHMH wants to achieve from its open source investment?”

For convenience, this framework groups potential open source goals into three categories:

1. *Development And Collaboration Goals*
2. *Outreach And Ecosystem Goals*
3. *Internal Goals*

Within those categories we cover a fairly long list of potential open source objectives. That list is not exhaustive, but if DOHMH plans to use open source involvement to achieve an outcome that is not listed below, that activity might be somewhat novel and would suggest an opportunity for input from an experienced open source practitioner.

For any given project, DOHMH should choose *at most three main open source goals* to pursue in the near term. Goals might change, and DOHMH might pursue a different set of goals as a project matures and as situations evolve. At any given moment, though, three main goals should suffice. While every item on the lists below might seem beneficial, and any of them can be pursued opportunistically, a strategy that tries to maximize all of these effects will be muddled and incoherent. An attempt to achieve all of these goals will result in failing to significantly reach any of them. As mentioned earlier, DOHMH is strongest when it acts with clear vision and focused purpose.

Although we describe a wide range of potential goals below, many of them are not suitable for the data portal. We include them here because this report is intended to support both the portal and broader generalized open source decision-making. Thus we present the entire list for reference.



Key Recommendation: From the set of open source goals, OTS recommends DOHMH prioritize three initial goals for the portal project:

- ★ *Breaking Vendor Lock-In*
- ★ *Improving Internal Collaboration*
- ★ *Framework For Partner Collaboration*

2.3.1 Development And Collaboration Goals

- *Framework for partner collaboration* — Open source is the cheapest, quickest, lightest-weight, least paperwork-laden, most nimble route to building collaborative solutions to problems that have in the past been tackled with heavyweight infrastructure like trade associations and multi-lateral agreements. Open source is well suited for multiparty cooperative efforts that need to easily add new participants who will play diverse roles that shift over time. Having a ready-made system for such collaboration is faster and requires less legal and logistical overhead than the alternative of recruiting partners and negotiating specific investment and structures with each of them.

While the open source world can move faster than the arrangements of old, there are times when collaboration will require more legal documentation and negotiated agreement than a simple Contributor License Agreement (CLA) and a standard open source license. For those times, there are available organizational homes that provide more consideration and legal assurances. The Fintech Open Source Foundation (FinOS), Eclipse Foundation (EF) and Linux Foundation (LF), for example, provide structures that are more like trade associations than, say, the Apache Software Foundation (ASF). Ultimately, projects can pick the type and weight of process they need and work within existing structures to enact it.

This ability to quickly and easily choose off-the-shelf governance structures frees DOHMH from some of the burden of supporting the organizational infrastructure a thriving project needs. Some projects do not need to be housed inside DOHMH, and there are benefits to having homes for projects that are better placed in the outside world than inside a New York City agency.



Key Recommendation: As DOHMH considers involving outside collaborators, develop an understanding of the range of project sponsorship organizations that might serve as eventual homes for DOHMH projects such as the portal.

- *Amplify, accelerate or expand developer base* — This is often the first goal people consider when they think about open source. Opening up development to additional

teams within DOHMH or extending cooperation to the wider community is the essence of free and open source software. At its heart, open source approaches are designed to reduce barriers to adding additional effort to a software effort. Whether that additional effort comes from DOHMH personnel working across functional areas, from additional vendors, or from external contributors, open source is a way to organize collaboration and establish common understanding. By pooling the work and avoiding duplication of effort, DOHMH can more effectively create technology that keeps pace with the rest of the field.

For government projects focused on serving immediate, internal needs, it is usually unrealistic to expect significant early participation from third-party developers who aren't directly hired by the agency. Open source prepares a project for working with such contributors, but it cannot make them appear. The portal has a community, peer agencies in the City, and potential partners in various state governments. The portal might eventually involve some of them in a significant way, but that is a goal that can be added later, after a variety of other concerns have been satisfied.

- *Lead a standardization effort* — Although much formal standards work still occurs, these days a good deal of coordination on protocols and Application Programming Interfaces (APIs) increasingly happens in open source projects, sometimes with their outputs submitted to standards bodies *ex post facto*. Free software collaborations provide an open forum for all interested parties to provide input. Their emphasis on proofs of concept and running code helps support implementations that serve as references or as real-world experiments. And the projects serve as useful places to solve patent issues as well. By creating standards and allowing anybody to implement them, open source projects drive standards acceptance. It is still true that “the nice thing about standards is that there are so many to choose from,”¹⁰ but standards proliferation is greatly reduced when rival teams can avoid competing with each other to see who will end up in control of a standard they can use as a chokepoint. Instead, open source allows an entire sector to rally around a smaller number of winners.¹¹
- *Replace an existing product* — Sometimes existing offerings fail to meet the needs of DOHMH’s mission or its stakeholders. Replacing incumbent products, though, can be difficult. FOSS changes the assumptions in an ecosystem, often by commoditizing products and turning them into base infrastructure. This produces a range of disruptive effects: Stakeholders come to expect open conditions. Entire fields of competitors can work together to replace incumbent products. Pricing models must be redesigned. Any or all of these effects can be useful in creating needed space for new solutions. By the same token, if you *are* the incumbent, making use of open source approaches can put you in control of these effects and minimize the opportunities for others to use these forces to disrupt your business model and market position.

One salient example of a public organization using open source to provide a better tool than the existing market offered occurred at the World Bank. They used a FOSS

¹⁰See Andrew S. Tannenbaum, *Computer Networks*, 2nd ed., p. 254.

¹¹See, e.g., the speed with which Docker was accepted as a *de facto* container standard.

approach to create GeoNode,¹² which gave a range of stakeholders a compelling option to replace one that was not meeting needs. GeoNode eventually became self-sustaining and today supports the work of hundreds of stakeholder initiatives around the world.



Further Reading: For more on GeoNode’s path from institutionally-supported software to community-supported open sources, see <https://opendri.org/wp-content/uploads/2017/03/OpenDRI-and-GeoNode-a-Case-Study-on-Institutional-Investments-in-Open-Source.pdf>.

- *Contextual insight* — There is a world of information hidden in open source projects. They reflect the priorities and approaches of an entire ecosystem of users, contributors and stakeholders. They contain leading indicators of priorities and actions. Few organizations make considered use of this information, but more of them should, and they should regard this information as a type of return on investment.

A number of well-resourced companies have looked into doing research on extracting actionable business insights from the ecosystem of open source projects. Whoever wins that race will have a unique window into the open source investments of competitors. This is not to suggest that DOHMH should enter this race, but it does speak to how much potential there is in the trails left by open source activity.

For DOHMH, it might benefit from systematic collection of information on who is engaging with a project and to what degree. This data can provide a lot of insights relevant to roadmap planning, resource allocation, and partner recruitment. Most FOSS participants rely on a vague impression of participation metrics, and this is a good start, but if understanding the ecosystem becomes a main open source goal, it probably makes sense to invest in more rigorous treatment of the available data.

2.3.2 Outreach And Ecosystem Goals

- *Break vendor lock-in* — Vendor lock-in is an increasingly significant factor in the procurement matrices of large enterprises and governments. Vendor lock-in is a huge risk because it positions the vendor relationship as a single point of failure in technology procurement. Multi-vendor ecosystems provide governments with the comfort of price competition, protection from the vicissitudes of a vendor’s shifting business model, and recourse to substitute suppliers. This approach reduces risk, which is often a primary consideration for government procurement officers. Software procurement officers prefer products that leave them many options and do not lock them down to any one source or platform.

In interviews with government agencies, OTS has observed that vendor lock-in is the most frequently-cited barrier to success in custom technology procurement. More and more agencies are turning to FOSS as a solution to this problem. By denying any one

¹²<http://geonode.org/>

vendor exclusive rights to a codebase, agencies retain the ability to turn to additional vendors and increase competition. More importantly, open source approaches allow an agency to foster a multi-vendor ecosystem that serves multiple jurisdictions deploying many instances and variants of a single open source codebase. That ecosystem becomes a growing value that resists lock-in across an entire sector of technology. Proprietary software (even proprietary software owned by a government agency) cannot break vendor lock-in as well as open source for the simple reason that it cannot grow multi-vendor ecosystems as well as open source can.

- *Engage with users* — Open source projects are open to the world, and that includes everybody with a stake in the software, all the way to end users. Open source projects have the opportunity to directly reach users to gather feedback, provide services, and be responsive. Large gains to product quality and user satisfaction can flow from there.

DOHMH's portal has a user community that includes people eager for more engagement. This includes the kinds of users that participate in the EPHT Portal User Group and recruit others to join up. They are candidates for greater involvement in the project. This is true even if these users are not experienced software developers. Successful FOSS efforts involve people of many different backgrounds and skills, and what matters most is building a structure that can turn enthusiasm toward productive participation. Although OTS does not recommend that DOHMH focus on user engagement as one of its early open source goals, we note that there is civic engagement potential here that the agency might pursue in opportunistic ways at any point during the project lifecycle.



Key Recommendation: Develop visible pathways for users of DOHMH's open source efforts to provide feedback.

- *Transparency for stakeholders and partners* — Open source collaboration happens in public forums. When an open source project is run well, most meaningful communication leaves a public trail, even if that trail is just an email summarizing a discussion or decision. Issue trackers, wikis, and code commits give everybody a direct window into the project at an exacting level of detail. This allows everybody to understand how decisions are made, see issues get prioritized, and understand how their own open source investments are paying off. To some partners this is crucial, especially in domains where FOSS is the approach that enables collaboration in the absence of well-aligned goals and philosophies.

For projects that involve government, public trust is as an essential component. Working in the open allows scrutiny, and as a result builds trust over time in ways that closed-door projects simply cannot. There are good arguments that the default mode for government software should be open source.

- *Establish a basis for project reputation* — Open source is a public display. If development and code are of high quality, it will be visible in the open code.

Documentation, process, and testing are verifiable. Security is open to inspection. For projects operating in fields where trust is a major concern, these qualities can be a useful differentiating factor.

- *Branding and credibility* — Open source is fashionable. People like it, and they like being associated with it. Appearing to be open source is so desirable that companies fight over when and how that label applies. One reason the label is valuable is because it can't be faked: following open source principles creates real benefits for everybody working in and around a FOSS project. Whether it is transparency or engagement or breaking vendor lock-in, investing the resources to live up to open source promises gives everybody a reason to stay engaged.

For DOHMH, this translates into reach and engagement that drives efficacy of its efforts. As openness becomes a reason to team up with DOHMH, mission goals can gain effective champions and resources from outside the agency's usual domain.

2.3.3 Internal Goals

- *Improve internal collaboration* — Open source collaboration practices have evolved to allow people to cooperate across entity boundaries, across timezones, and even across linguistic barriers. FOSS enables flexible cooperation between groups with differing interests. These are qualities that are useful even when a project is internal and not open source at all. The collaboration methods of open source have been spreading throughout the technology field. There is even an “inner source” movement that describes open-source-style practices confined within an organization's walls.¹³ Indeed, many organizations find themselves adopting open processes for tasks that have nothing to do with software at all.¹⁴

The portal team is largely internal to DOHMH, but that might change as a new round of development proceeds. Increasingly, development projects tend to pull in more diverse expertise from across a wide range of functional areas. As cross-team collaboration becomes the norm, so does inter-organizational cooperation. Open source is a strategy for improving collaboration internally that also prepares DOHMH for greater external collaboration as well.

- *Innovation* — Open source brings together a diverse set of parties, each of whom sees different possible uses of the base open source technology. Although DOHMH will tend to develop towards its own interests, marrying those interests to those of others in the ecosystem can produce new uses and make products valuable to additional stakeholders in new ways. Open source is not a magic source of automatic technological innovation, but it is a potent source of new ideas and approaches. The more DOHMH uses open source to cross-pollinate ideas with external stakeholders, the more powerful this effect will grow.

¹³See <https://innersourcecommons.org> for more information about innersource, including best practices.

¹⁴See RedHat's experience with this, detailed at <https://www.redhat.com/en/explore/the-open-organization-book>.

- *Improve developer hiring pool* — One of the most useful things about open source engagement is that it exposes DOHMH to a talent pool of people who have domain expertise in a technology that matters to DOHMH. It provides a public track record and a history of prior work with potential job candidates. New hires come in with confidence, having established relations with some DOHMH personnel, and can start their work with less time spent getting up to speed on the codebase and project. This is a large advantage for anybody who needs to recruit developers and technologists, and DOHMH's role in technology development is predicted to grow over time. Increased engagement with the open source project behind some of DOHMH's initiatives will yield hiring benefits, and can usefully precede increasing investments and headcount around those projects.
- *Improve morale and retention* — Developers increasingly want to work on open source products. They enjoy open source development methodologies and they also appreciate that domain knowledge about open source products improves their standing with their peers. Although in theory this means that open source helps them acquire transferable skills that are useful in the job market, in practice the result is often that developers value even more highly the jobs that provide these benefits in the first place. Some of the best developers will *only* work on open source for exactly this reason.
- *Improve open source capabilities* — Everybody who studies the technology sector believes free and open source software strategies will grow in importance over time. Organizations that can easily leverage open source approaches will be more effective than those that are confined to more traditional, proprietary approaches. One reason to invest in open source efforts is that as the technology world changes, DOHMH will need to have that experience in house. One outcome of DOHMH's open source work should be increased ability to execute a variety of open source strategies in future efforts.

Another benefit to investing in open source is that doing FOSS well can help non-open source, even non-software teams work smarter. Internal tools that are not themselves open source usually depend on a deep stack of open source libraries, and many of the resources in DOHMH's universe will be FOSS. Increased organizational capacity around open source will benefit anything that connects to open source, which increasingly includes almost everything.

2.3.4 Growing FOSS Capacity

To the list of recommended goals, OTS suggests one addition. Of all the goals listed above, DOHMH might add one that should be part of every FOSS effort DOHMH makes in the next several years:



Key Recommendation: In any FOSS work DOHMH undertakes in the near future, “Improve open source capabilities” should be an explicit goal. Policies and practices should be designed to maximize success of the project at hand but also to generate case studies and experience for informing DOHMH’s other FOSS efforts.

At a practical level, that means that DOHMH would benefit from conducting specific activities designed to analyze the results of engaging FOSS communities and to absorb lessons learned into widely-accessible institutional memory.

These activities can take a number of forms. For example, DOHMH can conduct reviews any of its open source work with a goal of capturing best practices that work well for the specific types of projects and constraints DOHMH faces. For an example of an organization doing this well, DOHMH might look at the World Bank’s report on its experience building mapping software for its extended community.¹⁵ DOHMH might also develop internal training materials that allow additional projects to take on specific categories of FOSS work (upstreaming bug fixes, building an open source community, contributing to an existing project, launching an open source project) while aligning with DOHMH policies and workflow. In that vein, some organizations have developed checklists to prompt consideration of the concerns that repeatedly arise during open source analysis and implementation. See Appendix A: “Open Source Analysis Checklist” for an example of what such a checklist might include. Another possible learning activity is to do community case studies that help an agency like DOHMH understand how best to approach its community and stakeholders.



Further Reading: For more information on how to conduct a community case study, see Red Hat’s Principal Community Analyst’s advice at <https://opensource.com/business/15/4/create-community-impact-case-studies>.

All of this learning and experience can culminate in materials that share knowledge and history. These materials can take many forms, including documentation, training materials, worksheets, checklists, and wikis. For learning that is specific to an open source project, we encourage placing them in the project repository or making them public in some other readily-available way.

2.4 Identifying Archetypes For Best Practices

Over the past couple decades, open source projects have settled into a set of recognizable patterns or “archetypes.” These archetypes shape goals, licensing, development style, collaboration style, and many other aspects of a project.

¹⁵See <https://opendri.org/wp-content/uploads/2017/03/OpenDRI-and-GeoNode-a-Case-Study-on-Institutional-Investments-in-Open-Source.pdf>.

This section describes how to use archetypes as a tool for pattern-matching to gain insight into existing or planned open source investments. By this point, you should have clarified big picture goals, gathered information about our ecosystem, and picked open source goals on which to focus. From there, we can look for patterns found in the open source world that might inform the portal’s work.

Open Tech Strategies made an initial survey of archetypes in “Open Source Archetypes: A Framework for Purposeful Open Source.” A complete discussion of each archetype is available in that report; below we list their names, from which one can draw a rough idea of the project pattern each one represents:

- Wide Open
- Trusted Vendor
- Controlled Ecosystem
- Multi-Vendor Infrastructure
- Upstream Dependency
- Rocket Ship to Mars
- Specialty Library
- Mass Market
- Business-To-Business (“B2B”) Open Source
- Bathwater



Further Reading: Published with the Mozilla Corporation, the Field Guide To Open Source Archetypes is available at https://blog.mozilla.org/wp-content/uploads/2018/05/MZOTS_OS_Archetypes_report_ext_scr.pdf.

DOHMH will encounter all of these archetypes in the FOSS projects it deals with. Familiarity with the archetypes will help DOHMH personnel understand those projects and get the most from working with them.

These archetypes are not rigid boxes. Whether DOHMH originates a project or encounters one of these archetypes in the field, some projects will fit cleanly into an archetype better than others will. Indeed, very few projects fit one single archetype *perfectly* — other archetypes often offer additional illumination.

Furthermore, projects transition between archetypes over time. At its start, a DOHMH project might start out as a “Trusted Vendor” effort leveraging DOHMH’s position in the world as a credible and beneficial actor. Over time, though, increased involvement from partners and other jurisdictions might eventually produce a Controlled Ecosystem that maintains a central core while an ecosystem of customizers, integrators, and deployment specialists help translate open source value into real-world impact.

Archetypes analysis is useful even for projects that start out with a goal of remaining internal to DOHMH (i.e. “innersource” projects). Those projects will gravitate towards Wide Open models, at least initially, and some of the practices of Wide Open projects are particularly useful to innersource efforts. For example, DOHMH’s work might include support for internal onboarding that helps new personnel and new teams join forces with

existing innersource work. That investment makes it easier (and thus both more likely and more effective) for innersource partners to work together, which increases the chance of successful outcomes for DOHMH as a whole.

The purpose of placing one's project within the space of open source archetypes is *not* to "correctly" categorize one's project — that would be an empty exercise. Rather, the purpose is threefold:

1. *To suggest questions about the project that would be useful to ask early on;*
2. *To suggest activity patterns to look for as the project progresses, and thus to enable one to be better prepared to handle and take advantage of those patterns;*
3. *To anticipate likely future directions of evolution for the project, again to be prepared for — and even to consciously work toward — those changes.*

Done right, it is the process that matters, not the specific outputs. Attempting to identify a project's most suitable archetype is usually as illuminating as the answer one finally lands on. A sample of the questions that go into identifying an archetype include:

- Where participants come from (e.g., from which parts of DOHMH, or from the user base, or from other institutions who have similar needs, or from other institutions who have orthogonal needs, etc.), their motivations (e.g., their open source goals), and the kind and amount of resources they will contribute;
- What will the project count as success: widespread adoption, technical achievement, meeting a particular business goal, undermining a competing product, or something else?
- The pace of development;
- Ease with which a newcomer becomes a regular contributor (note this is distinct from the previous question);
- The technical design of the project
- (how modular is it, what other technologies is it intended to work with, etc);
- Amount of attention and overhead devoted to incorporating new contributors into the project;
- What kind of open source licensing is most appropriate and why;
- How are decisions made in the project right now? How should they be made a year from now? Five years from now? (In general we do not advise spending a great deal of time formalizing governance early in a project, but observing how the project's governance works in practice at a given point in time is still useful for understanding what kind of project it is.)

This is not a complete list of questions to ask. Reading through the available archetypes and the assertions made about each one will help one formulate the questions most useful to a given new project.

To take advantage of these archetypes, while at the same time not being unduly constrained by them, DOHMH should:

1. Look for questions that ring true or seem especially useful for *DOHMH's* work;
2. Pick the archetypes that apply (there will often be one primary archetype, but look at others as well to find other relevant elements);
3. Compare with the example projects given;
4. Describe DOHMH's project to its participants and stakeholders in terms of a primary archetype with customizations — the variant parts will illuminate what is special about the project's environment and goals.
5. After careful consideration, discard any aspect of an applied archetype that doesn't fit the project and its context.

The ways in which a project doesn't completely match with the primary identified archetype are not indications that the project is doing anything wrong. One should always expect to depart from the archetypes to some degree. A project doesn't succeed by pursuing a predefined open source strategy, but rather by forming a strategy that matches its goals and operating environment. The purpose of identifying an archetype is both to make the process of defining that strategy more efficient, and to make the strategy more likely to work because it incorporates techniques that have worked in other similar projects.

When considered in enough detail, every successful project is unique and none wholly conforms to the ideals described in an archetype. Successful projects are always well adapted to their niches. They meet specific needs in unique ways. For this reason, we expect every project to deviate from the archetypes in ways both large and small. This is healthy, and understanding *why* a specific project deviates is more useful than trying to force conformity with any archetypal quality.

3 Embracing Open

As the portal moves toward an open source future, there are a variety of mutually reinforcing steps that can be taken to support a development process that yields tangible open source benefits.

Again, for the portal project OTS recommends the following set of initial open source goals, as discussed in Section 2.3:

- *Improving Internal Collaboration*
- *Breaking Vendor Lock-In*
- *Providing A Framework For Partner Collaboration*

It might seem odd to recommend that the portal project adopt practices honed over decades of optimizing distributed collaboration across boundaries. After all, the portal’s development is currently a largely internal project. Even if it aims to invite external partners, it will take time for those partners to show up and make significant contributions. Given a public agency’s need to justify its resource investments, it is important to be able to articulate the near-term value that open source provides.

The reason open source practices work internally is that even if DOHMH does not engage in open source outreach, responsibility for the portal will be spread across a variety of entities. Even just within DOHMH, different members of the team operate on separate concerns, hand off work to each other, and make decisions in subsets of the larger group. This is why in many organizations, even projects that have no external partnerships often still adopt open source practices, sometimes calling them “inner source”¹⁶ or “open organization”¹⁷ reforms.

Moving beyond collaboration among portal team members, there are other collaborators within DOHMH as well as City partners just outside DOHMH’s gates. The Intergovernmental Affairs Group, for example, is, in the words of one interviewee, “a major user.” Likewise NYC Opportunity and BetaNYC are eager to collaborate on building a city that takes advantage of the data-driven benefits promised by the civic tech movement and human-centered design. There are also opportunities to build tighter relationships with the City Council’s data team, which can help DOHMH better inform City policy and invest in relations with the Council. Throughout the City, there are agencies that could use better access to DOHMH’s data, or use its infrastructure and methods for pursuing data-based engagement with New York’s citizens.

Together, the starting goals and an environment filled with potential partners paints a picture of possible scenarios DOHMH might achieve by investing in open source approaches. DOHMH might start by moving its development practices and infrastructure toward modes that enable distributed collaboration across organizational boundaries. That is, DOHMH might increase use of agile development methodologies to manage the portal process internally and with any contractors or software development vendors it brings aboard. It could, for example, apply internal resources where its domain knowledge is most valuable: at the design, requirements, and prototyping stage. The internal staff who did that stage of work could then use open source collaboration models to work closely with external teams to implement those designs. This use of FOSS process would also allow DOHMH staff enough touchpoints in the vendor development cycle to keep vendors aligned with the vision in those designs. This would address a pain point found in past work with development vendors. Over time, it would also yield a project that is well-constructed for expanding external engagement.

Adopting open source collaboration methods would also enable DOHMH to engage vendors and contractors who, together, form a multi-vendor ecosystem that resists vendor lock-in. This approach, which the federal Government Services Administration (GSA)’s 18F calls

¹⁶See <https://innersourcecommons.org/>.

¹⁷See <https://opensource.com/open-organization>.

“modular contracting,”¹⁸ involves a constellation of vendors working together on a larger software development effort. The only way to manage a project like that is to adopt practices that enable distributed development across organizational boundaries. Any other approach is too likely to fail.

Those same methods become an easy, flexible way to invite participation from a wide range of potential partners. As a whole, the system might serve other agencies concerned with environmental health data — namely peer agencies, mostly at the state level.¹⁹ Individual pieces of the portal might be useful to other types of participants. Data visualization widgets might be useful to other agencies, scientists, and journalists. Standardized ways to manipulate, clean, and distribute datasets would be useful in many industries. Data science is a young field, and there are many places that need improvements in workflow support for authoring data-backed articles, integrating Jupyter notebooks, and reviewing or editing articles. Any or all of these aspects of a new portal might lead to open source engagement if those aspects are modularly accessible to external partners.

As soon as DOHMH finds traction anywhere that leads to additional deployments of all or part of the portal’s codebase, it can begin to assemble an ecosystem. Vendors will show up to meet the demand, and they will also work to spur more adoption. Other jurisdictions will have confidence they might find the codebase useful. Some might even contribute code back, if only to keep their own version from slowly falling out of alignment with updates to DOHMH’s work. These are the beginnings of a cycle of cooperation that gains value as more participants join in. Those new participants contribute additional resources to the ecosystem, which makes everybody’s work more valuable, and hopefully this feeds a virtuous cycle that attracts future rounds of participants.

These three goals thus work together. By moving in open source directions, nurturing an ecosystem of vendors, and providing a nimble, low-friction framework for inviting participation, DOHMH can take small, considered steps toward open source cooperation with a wide audience. Even if that audience does not emerge, though, DOHMH still gains all the benefits of improved collaboration within the agency and with peers across the City.

Together, these goals combine to suggest a strategy in which DOHMH adopts a “lazy” approach to Wide Open: create wide open structures that invite participation from a wide range of potential collaborators, but without strong opinions about where that collaboration will emerge. Partners might come from within DOHMH, from other agencies, state agencies across the country, the federal level or industry. The portal might even see interest from engaged citizens, students, and academics. It would be a mistake to narrow in on any one aspect of the project to the exclusion of others.

In a lazy Wide Open project, one would expect the project to prioritize building structures that accept and foster open source collaboration over initiatives that extend outreach to potential develop communities. This approach accepts that it is not yet clear which communities are most likely to engage with the portal on a development level (as opposed

¹⁸For more on modular contracting, see Section 4.1.

¹⁹It is possible that as the cost of collecting and processing data drops, other municipal jurisdictions might be interested as well, but that interest is perhaps too speculative to include in planning at this stage.

to, say, on a data consumption or policy basis) and so the project's best growth strategy is to be prepared to make the most of whatever contact does come in. As soon as it has more information about promising potential contributors, it can conduct focused outreach to convert that potential into actual participation.

In addition, BESP will have other opportunities to embrace open source more broadly than in just the portal. First, BESP will use a lot of open source in the course of its work, both on the portal and, increasingly, throughout the bureau. This work will allow BESP to provide examples for government about how these projects can be used to solve common problems. One recent example is the use of the Vega-Lite data visualization project to create the mapping component of the agency's COVID-19 dashboard. By working in the open on GitHub, DOHMH can serve as an example, providing documentation and use cases by learning in public.

That type of generalized open source engagement can lead back to the portal. It raises DOHMH's open source profile with other agencies who are the likely target audience for portal participation. It also connects BESP to related projects in the wider govtech and data community. Those connections are the starting point of community for future DOHMH work.

3.1 Public Communications

To suggest that the portal might adopt a lazy Wide Open strategy is not to advise against pursuing current outreach opportunities among portal users. Outreach should be a major component of the portal work because increased engagement with the portal's data and infrastructure is in and of itself on-mission for DOHMH. Generally speaking, though, the aim of that type of outreach is not specifically oriented on recruiting participation in the portal as an open source project. Rather, that type of end-user outreach focuses on increasing user engagement with the content of the portal. In that sense, potential user outreach differs from potential developer outreach.

The portal team already does a fair amount of user outreach. It works with the EPHT Portal User Group to determine user needs and workflows. Insights from this user group inform design and prototyping. Involving the community in the design is a form of outreach. Those people tend to become ambassadors for the project, to tell their colleagues about the portal, and some might be candidates for other types of increased engagement over time.

3.1.1 Instructional Videos

One potentially effective investment is to produce short instructional videos that explain how to use various aspects of the portal. Such videos do not need especially high production values; if they are merely competently done and engaging, and are publicized effectively, that is enough. What matters is that they exist and cover functionality people are interested in.

In our experience, three to four minutes is the right length for such a video, and it is important that any descriptions of or links to the video emphasize its short length, so that potential viewers understand in advance how small an investment they would make by deciding to watch it. The New York City Office of the Comptroller has had some success engaging users with instructional videos that teach the public how to make more sophisticated use of the Checkbook NYC web site.²⁰

3.1.2 Consult with Other Departments and Organizations

There are many possible investments DOHMH might make in reaching new users, and DOHMH already does significant work in this area, especially in reaching out to students. It might also consult peers at other City departments or organizations that are historically good at community outreach, like the American Red Cross.²¹

While the rest of this section is about outreach designed to increase open source participation, increased user engagement is also an important component of project growth. Without usage, no amount of open source investment will attract significant partner participation.

3.2 Open Source Infrastructure

OTS recommends that the portal adopt at least the minimum set of collaboration tools typically relied upon by open source projects. Without these tools, it will be difficult to operate an effort that behaves like an open source project. That minimum set includes these pieces of collaboration infrastructure: version control, issue tracker, mailing list, and perhaps real-time text-based chat as well. These tools are well covered in chapter 3 of Karl Fogel's book *Producing Open Source Software*. Below, we touch briefly upon each tool and then link to specific passages in that book.



Further Reading: For any organization that plans to participate in operating an open source effort, Karl Fogel's book, *Producing Open Source Software* is worth reading in its entirety. It is available online at <https://producingoss.com>.

- Version Control — The portal needs a public version control repository. The project might debate policies on when to place which material under version control, but there is no question it needs a source control repository. Base portal code belongs in a repository, but so do widgets as well as research and editing artifacts.


²⁰<https://www.checkbooknyc.com/instructional-videos>

²¹In particular, it might be useful for DOHMH to speak with Jim McGowan, the Director of Information Management & Situational Awareness for the Chicago and Northern Illinois Region of the American Red Cross, who has a particular aptitude for this kind of outreach and has experience coordinating it with open source software efforts.

Currently, too little of the work produced for publication is under version control. That includes the code written for widgets as well as stories themselves. Editing artifacts are not preserved in any systematic way, nor are they available to the entire team. Different contributors work in different modes, so it would not be feasible to force everybody into a common authoring environment. Still, even short of that step, channeling everybody toward a repository is a fairly easy adjustment to make.

Ideally, the new portal workflow will deploy widgets directly from this repository. That is, using the repository will be on the direct path to publication. This is important to ensure its use and also to give users many chances to use (and thus learn) the repository. Of course, repository management will require support and training.

For DOHMH's purpose, we recommend choosing a git-based repository. It is a reasonable default and git skills are common among open source developers. Another reasonable default is to host this repository on GitHub.²² Please see <https://producingoss.com/en/producingoss.html#vc> for more information about version control tooling for open source projects,

 **Key Recommendation:** Create an open source repository for data visualization widgets, preferably hosted on a public server that invites widespread interaction (e.g., GitHub). Provide training to DOHMH staff in its use.

- Issue Tracker — There aren't any easily-managed feedback mechanisms for the portal, either for the public or for staff use.


It appears that public feedback arrives via ad-hoc emails from users. Internally, feedback appears to flow through the portal team well enough,²³ but this too is informal. There are no mechanisms to track feedback and responses. That is, there is no ticketing system for addressing issues that affect users or that have implications for the software and widgets. This is common for legacy projects, but not for modern, open source efforts.

This is not to suggest that DOHMH should establish a support desk for end-user issues. Rather, we recommend that as public feedback arrives, whether via the tracker or less formal channels, portal staff manage the internal response (e.g., to investigate or remediate issues) in a structured way. Since the portal will be using an issue tracker for technical issues, it makes sense to place user-identified technical issues there as well, with the understanding that these are issues to fix within the project, not support requests for users.

Please see <https://producingoss.com/en/producingoss.html#bug-tracker> for more information about the role issue trackers play in open source projects.

²²If DOHMH finds itself needing guidance on repository workflow, OTS can provide materials developed for other public entities on the topic.

²³This was demonstrated during OTS's research in the many instances of members of the portal team accurately representing the views and the work of their teammates.

 **Key Recommendation:** Establish and use an issue tracker to manage workflow related to technical tasks and bugs.

- Message Forum / Mailing List — Currently, DOHMH’s communication about the portal follows patterns typical of internal projects. The project delegates different concerns to various staff, and each works within his or her domain, coordinating with others as needed, usually by person-to-person email or by voice conversations.

In open source efforts, by contrast, communication flows are less ad-hoc and more group-oriented. Projects have evolved a set of standard practices that ensures that everybody can access the information they need to contribute, make decisions, or just stay up to date. The primary way to do that is via a mailing list or message forum (hereinafter “mailing list” for brevity).

The mailing list acts as institutional memory, which is useful for distributed projects that, by their nature, rely less on tight connections to centralized structures. This is not to say that every decision should be debated on the list. When possible, high bandwidth, face-to-face discussion is still a better medium than email for making decisions. Rather, the mailing list is the place where those individual discussions are officialized into project decisions.

Posting messages that summarize decisions and briefly describe the reasons for those decisions allows an open source project to keep everybody aligned. The goals of such messages are to inform and to record, not to convince. If there is still convincing left to do, the decision has not been reached.

Once recorded, these decisions form the official record of the project. They become the authoritative basis for future discussions and help avoid retreading old ground as new participants join the effort. For current, internal collaborators, the mailing list archives are a way to onboard new employees and update current staff.

Please see <https://producingoss.com/en/producingoss.html#message-forums> for more information about how open source projects use mailing lists.

 **Key Recommendation:** DOHMH should set up a mailing list or forum to act as a clearinghouse for development decisions.


- Chat — If the mailing list is the place where projects record decisions, real-time chat is often where they conduct the discussions that lead to those decisions. For teams that are separated by geography or timezones, real-time chat is a vital social element of the project and helps erase the line between “present” and “remote”. The chat is where participants build trust and make human connections, which are vital parts of creating teams that can manage the conflict that inevitably arises in distributed teams composed of participants with varying aims. Whether DOHMH chooses

Slack,²⁴ Zulip (a commercial, open source competitor to Slack), or some other tool, there is no replacement for real-time communication within the project.

A public chat service would also be of great benefit as an outreach channel. The immediate ability to directly reach the portal team enables interactions to happen when they are most at the forefront of somebody's mind. The delay involved in other mechanisms like email often results in interest waning before the project has a chance to engage. Even worse, many users who might be willing to drop into a chat room will not engage via email at all, which means the project might miss them entirely.

Despite the advantages listed here, of the four collaboration tools mentioned in this section, chat is the lowest priority and might not be a necessary component of team interaction. That is not to say DOHMH should avoid it, just that it is a lower priority than other tools.

Please see <https://producingoss.com/en/producingoss.html#online-chat> for more information about real-time chat systems for open source projects.

 **Key Recommendation:** DOHMH should consider adding real-time chat to its collaboration toolkit, both for portal teammates as a way for the public to reach the team directly from the portal.

Each of these items is a key enabler of agile collaboration. No modern software development effort can proceed efficiently without at least this set of tools. On open source teams, which are specifically designed to enable wide collaboration across boundaries, these tools become more important. They are necessary substitutes for hallway conversation, face-to-face meetings, and high cadence contact.

What's most important about using these tools is that they are available to the portal's development community. If that community includes other City agencies, staff at those agencies should be full participants in the project's collaboration mechanisms. If development is open to outside vendors, staff in other jurisdictions, or the public at large, these tools must be deployed so that each of these groups can join the conversation. Even if development is open to a relatively small circle today, anybody whom it might widen to include should have at least theoretical access. The conversation should proceed on an open basis from the start, if only because the overhead of opening a closed conversation (especially reviewing its history for material that isn't appropriate for publication) often prevents conversations from opening at a later date. The adage that projects should be "open from day one" applies as much to the communications infrastructure as it does to

²⁴Slack is, notably, a proprietary tool. Some open source developers and communities avoid it for that reason.

the codebase.



Further Reading: For more about being open from day one, see the OTS blog post at <https://blog.opentechstrategies.com/2018/09/be-open-from-day-one/>.

Adopting these tools reduces the friction involved in bringing new staff and new vendors on board, including bringing them in to develop just one component, module, or aspect of functionality, user-facing or otherwise. Furthermore, these steps makes it possible to point to current code and other development artifacts directly from an RFI or RFP (see Section 4.6), thus enabling vendors to self-filter for interest and competence.

In addition to the above tools, there is an entire universe of “agile” tools designed to support modern development teams. Some of these tools are quite useful, but none of them is universally necessary in the same way as the tools listed above. DOHMH might explore the universe of agile-oriented tools and adopt the ones that produce results, but it should start with the core set of tools we have identified here.

4 Open Source Municipal Procurement

At this point, OTS has described an open source approach to building a new portal and sketched out some of the details on what DOHMH might build and how to build it. There is one aspect of delivering a revised portal still left to cover, which is how to execute on this plan within the bounds of a city agency.

FOSS has rapidly become successful in the commercial sector. Few modern enterprises can thrive without taking full advantage of open source antecedents to power a competitive business. More and more sectors find themselves taken over by open source offerings that crowd out once-dominant proprietary competitors. It is fair to say that open source, once maligned by much of the commercial software world, has emerged victorious and is now a major component of every software effort of note today.

Relatedly, an agile, iterative, and continuous style of development has taken hold at the most successful technology companies. Gmail, Facebook, and other online services are constantly being rewritten and redeployed, on a component by component basis, at a frequency measured in days and sometimes even in hours. Companies have learned how to structure their software development efforts so that user-facing systems can be rebuilt continuously, partially, reliably, and efficiently. A change in one place doesn't make everything else break, because clear module boundaries have been defined. These

boundaries often reflect organizational divisions as much as technical ones.



Further Reading: The US Government Accountability Office published a guide in 2012 that is a useful resource for government agencies engaged in agile development: <https://www.gao.gov/assets/600/593091.pdf>

This style of development is closely connected to open source: the availability of swappable open source components makes modular development not only possible but practically inevitable, and it rewards companies for participating in the open source projects they depend on the most.



Further Reading: For a good introduction to modular architecture, see this blog post, which is the first in a two-part series: <https://medium.com/on-software-architecture/on-modular-architectures-53ec61f88ff4>. Another useful resource is <https://blog.fedecarg.com/2008/06/28/a-modular-approach-to-web-development/>, which focuses on web development.

Generally speaking, though, government has lagged behind. While governments have begun to embrace open source approaches, actual, open source successes are much more rare in government than in the private sector. There are a lot of reasons why this is the case,²⁵ but none of them is that open source is somehow unsuited for government use. Rather, government has not yet spent a decade honing open source practices that take into account the particular needs of government software development.

One place this disparity appears is in contracting. Government has an obligation to safeguard the public trust. Its operation is accountable to public political process. Public investments are judged by different metrics than those used in private-sector commerce. While government has much to gain from open source and is in the process of developing and spreading best practices on how to do that, it cannot simply borrow procurement models wholesale from the private sector.

That is not to say that government lacks the knowledge or ability to succeed at FOSS projects. It does suggest, though, that approaches should be designed with care for the particular constraints and opportunities found in public agencies.

4.1 Modular Contracting

One center of excellence in government development of FOSS software is 18F, the digital service delivery arm of the federal GSA. They focus on technology procurement for federal agencies, and their methodology typically begins with agile and open source approaches. They advise federal projects to consider procurement based on a model of “modular

²⁵<https://producingoss.com/en/producingoss.html#governments-and-open-source> discusses some of those reasons.

contracting,”²⁶ in which large projects are procured via a set of small contracts, each to the vendor most suited for a particular task. This is a good starting point for state and local technology procurement as well.²⁷

This approach has several benefits, beginning most importantly with breaking vendor lock-in. At any point, a well-procured effort should provide DOHMH with recourse to multiple, credible vendors, each of whom has familiarity with the software, experience working with the other teams, and is able to work well with the agency’s open source approach. Every vendor becomes replaceable because none by itself is so crucial to the process that it cannot be replaced.

With that goal in mind, 18F and others in the industry recommend structuring procurement as multiple contracts distributed among a group of vendors. They also recommend relatively small contracts, though “small” often refers to a different scale at the local level, even in a locality like New York City.

It is difficult to make one-size-fits-all recommendations as to the correct size of a modular contract. That will depend on the overall budget, the project, and the ecosystem needs at a particular stage of development. For the portal project, OTS recommends contracts no smaller than \$60,000. This size will allow multiple vendors while still providing enough budget room for vendors to deliver without squeezing margins so much that quality suffers. Generally, on the upper end, OTS does not recommend a monetary limit— some modules will, after all, make the most sense as larger components. The key is to choose module sizes that align with both the component and the overall multiple-vendor ecosystem DOHMH is building.

One reason to structure procurement this way is to reduce risk associated with any one vendor. There is a lot one might say about the nature of risk in government technology procurement. For the purposes of this report, though, spreading risk among a series of vendors reduces the chances of catastrophic failure and increases the ability to recover from problems, while still allowing an agency to concentrate overall responsibility for delivery in the hands of one integration vendor. This reduction in risk is usually worth the attendant increases in cost and contracting overhead that are inevitable when one is working with more vendors.

One of the ways this strategy improves the risk outlook is by opening options for DOHMH to contract with more-focused vendors who have specific expertise. No one development firm is good at every possible technology stack. Native Android and iOS expertise differ from each other and from web development. Many development vendors are good at one of these; few can deliver all three at the highest levels. Similarly, front end differs from back end development and they both depart from user experience design. Small contracts — typically under \$100,000, in this context — allow DOHMH to choose the best vendor for a given task, not one vendor who averages out to be the best across the board. This allows DOHMH to hire subject-matter experts and where needed to access the best experience

²⁶See <https://18f.gsa.gov/tags/modular-contracting/>.

²⁷18F addresses non-federal modular contracting at <https://18f.gsa.gov/2016/11/15/modular-procurement-state-local-government/>.

industry can offer. In any event, DOHMH might keep in mind the city threshold for an “M/WBE Noncompetitive Small Purchase” is \$150,000.²⁸

Issuing small contracts might also help agencies meet City targets for contracting with small and minority-owned business. OTS’s experience after interviewing many government procurement officials is that a major barrier to small business and MWBE contracting is a perception of risk in asking a small or new business to tackle a large, important project — established vendors often end up with the big contracts partly *because* they are established. Smaller contracts are inherently less risky, and multiple vendors working in an open mode makes it likely that issues will be surfaced before they become catastrophic, when remediation might still be easy and relatively inexpensive.

Finally, small, modular contracts align well with a technologically modular design. Modular contracting enforces modular technical design as different teams need to separate concerns so they can all develop in parallel. Similarly, a technically modular portal is susceptible to development by a variety of smaller vendors in ways that a monolithic structure would prevent. While DOHMH *could* hire one vendor to build a modular portal, teams that work together under one roof are more likely to — indeed, almost inevitably will — violate modular boundaries under deadline pressure. Separating those teams helps enforce the technical boundaries that keep open source process functioning.²⁹



Key Recommendation: Modular contracting works best with agile, open source development and modular technical design.

One aspect of modular contracting that is sometimes overlooked is that it can be costly to conduct many smaller rounds of procurement. Even if those smaller rounds qualify for faster processes with less overhead, the logistics around each agreement are themselves a factor that introduce both delay and risk to a project. It is quite common for all parties to agree on a set of work to be done and yet never proceed all the way through the contracting stage for various reasons.

Vendors too see a rise in costs. Because municipal scopes and budgets are often smaller than for federal projects, vendors at this level of government often find the small sizes of modular contracts a challenge. Smaller contracts for shorter periods of work provide less stability. Vendors find themselves unable to make the long-term commitments needed to hire long-term employees instead of ad-hoc contractors tend not to stay as long. That raises their costs, which of course eventually raises costs for DOHMH.

For these reasons, OTS suggests that agencies engaged in modular contracting place those contract modules in larger Master Services Agreements (MSAs) while also encouraging vendors to seek multiple contract modules under their MSA. In addition, DOHMH should increase the size of modules as a project proceeds. Once a project has a set of vendors who understand the product, demonstrate open source expertise, and have established a track

²⁸See <https://rules.cityofnewyork.us/content/mwbe-noncompetitive-small-purchase-provisions>

²⁹Another approach to enforcing modular separation is through OSQA, described below in Section 4.3.

record of delivering quality work on time, the benefits of modular contracting decrease while the costs remain elevated. At that point, DOHMH might opt for larger contract modules, especially as the project grows in scope. It might also allocate these larger modules among its existing set of vendors as a way to balance risk management with cost management.



Further Reading: For a comprehensive view into modular contracting, there is a 2012 guide produced by the Obama administration: <https://obamawhitehouse.archives.gov/sites/default/files/omb/procurement/guidance/modular-approaches-for-information-technology.pdf>.

4.2 Intellectual Property Contract Terms

Beyond modular contracting, there are a series of best practices that can help DOHMH maximize the benefits of its vendor ecosystem. We often start that discussion by considering contract terms.

Many government software vendors have traditionally retained intellectual property rights to the software they create, even though that software was created at public expense. There are a variety of reasons why such arrangements were common in the past, but many of the conditions that made it necessary no longer hold. For example, government software is no longer likely to be delivered as a monolithic stack in which the government-procured layer is but one small party tightly coupled to a variety of proprietary parts owned by third parties. Modularity, FOSS libraries, and standardized, open stacks make that architecture a thing of the past.

Perhaps more importantly, government procurement agents today have more options in the marketplace. There is a growing number of experienced vendors capable of delivering world-class software who don't require exclusive legal rights to exploit the codebase at the end of the contract. Open source is competitive, and is driving wholly proprietary approaches out of the market.


There are three classes of software delivered in a typical procurement scenario: third-party open source software, pre-existing vendor-created software, and code custom-written by the vendor for the current project. Notice this list does not include any third-party proprietary software.



Key Recommendation: Contracts should expressly forbid satisfying any deliverable with software that includes any proprietary component.

These different types of software are to some degree distinguishable for procurement purposes, but might be intermingled in the source code. We distinguish them at contracting just to ensure DOHMH has the rights it needs to proceed with its open source

plans and never needs to secure a vendor's permission to operate, improve, or hire other parties to work on its software. This is the key point.

 **Key Recommendation:** No matter what happens with intellectual property rights at the contracting stage, DOHMH must have the ability to deploy, distribute, and modify the software under a suitable open source license.

For third-party open source software, this means that DOHMH receives that software, clearly labeled, in a manner compliant with their original open source license and under terms that are compatible with the intended license of the final product. For code written by the vendor for other engagements and not paid for by DOHMH, this means delivered to DOHMH under open source license that allows redistribution under the intended license of the final product. There is no need for DOHMH to gain exclusive rights to these classes of software.

For code written by a vendor and paid for by DOHMH, the question of who should end up owning the rights is up for debate. Some agencies might be willing to see those rights remain in the vendor's hands as long as they receive an open source license that allows further distribution under the intended open source terms of the final product. So long as DOHMH secures that, it does not much matter who holds the copyright. Common practice, though, appears to be that the contracting agency ultimately gains all rights. In some cases, the vendor might receive a license back allowing proprietary relicensing so they can resell that software to other markets. A vendor might even enjoy a period of exclusivity in its ability to make proprietary use of the software. Specific arrangements will vary, but OTS sees no harm in them so long as a) the terms serve a larger goal of fostering a multi-vendor ecosystem, and b) DOHMH always has full rights, including redistribution rights, under the desired outbound open source license.

The County of Los Angeles, for example, engaged a vendor, Smartmatic, to build new open source voting machines that will debut during the primary elections in March of 2020. LA owns the resulting software and hardware designs, and Smartmatic enjoys a period of exclusive ability to use proprietary licensing to exploit the designs in the market. At the same time, LA is moving toward granting the public open source access to these materials. Smartmatic will have the proprietary rights it wants as it tries to sell the system in other jurisdictions. While it will have to compete with the open source crowd, it is welcome to build a proprietary business if it can do so under such conditions. Officials in LA hope this will help create the demand that entices more participants to join the effort. Other public-spirited projects in which a primary vendor predominates have either considered or adopted similar arrangements.

It is worth making one final point about rights. Open source licenses commonly deal with copyrights. They do not adequately address trademarks and are uneven in their handling of patents. Contracts must secure terms that prevent a vendor from encumbering further development and distribution on either trademark or patent grounds.

4.3 Open Source Quality Assurance

In addition to intellectual property clauses, OTS recommends using procurement structures to emphasize open source development *process*. A vendor who merely delivers a timely set of open source components has not actually done enough to succeed at contributing to a successful open source project. In addition to writing quality open source code, the vendor must enable all the other teams to succeed by participating in and reinforcing the open source process.

Some vendors will be unable to manage this participation. They won't have the open source experience or, in some cases, the temperament. No matter how much DOHMH might prefer experienced open source vendors during the procurement process, it is likely that at least some vendors will end up on the team because they submitted bids that scored high on non-open source criteria. To succeed, DOHMH cannot assume all its vendors will have the open source skills they need. DOHMH can instead provide structures that ensure success even when relying on vendors who might otherwise fall short in their open source practice.

Similarly, some vendors will be *unwilling* to participate in good faith. That is, they will be willing to check boxes on the paperwork, but behaviorally will default to the type of closed development process typically found in government contracts. Sometimes this unwillingness comes from a lack of experience and open source skill. In some other cases, perhaps, a vendor may hope the open source process fails, which could result in a final product that, while technically open source, cannot actually be released as open source, thus leaving the vendor as the sole source for that component in the government procurement marketplace.

Whatever the reason for failure to participate in open source process, DOHMH must have a way to bring recalcitrant vendors into process compliance. Otherwise, the project risks failure across multiple teams. To prepare for such situations, OTS recommends instituting OSQA at the project management level.

OSQA is a set of practices designed to ensure that DOHMH's open source project *behaves* like an open source project. If vendors are delivering open source code but not engaging in the process, the portal will hit some of its milestones but fail as collaboration falters over time. Adding new vendors will be difficult. Teams will discover API mismatches when they try to integrate modular pieces. Patches will go unreviewed, or worse, be accepted anyway. Documentation will degrade to the point that modules can only be deployed by the teams that developed them, which reintroduces vendor dependency and lock-in. By the time these failures are obvious to DOHMH, the portal will struggle as it experiences all the complexity of modular contracting without the benefits.

There are a variety of OSQA techniques DOHMH can introduce. OSQA "sits on the tree" and enforces standards at the pull request stage. This means OSQA approval is needed before source code changes can be incorporated into the project. Because incorporating changes to source code is a required deliverable for all software development vendors, when OSQA sits on the tree, it prevents vendors from fulfilling contractual milestones unless they meet quality standards. Vendors soon realize they cannot invoice if they do not

deliver approved code, which provides OSQA an entry point into a collaborative discussion about best practices that deliver high-quality code that passes approval standards quickly.

OSQA enforces policies about testing, adherence to design guidelines, accessibility compliance, and communications (e.g., as using the public bug tracker rather than private emails), and it reviews documentation. OSQA periodically attempts to build and deploy by assigning a test deployment to an operations staffer who has no prior knowledge of the software. If that person, armed with nothing more than the documentation, cannot deploy, that surfaces issues that must be addressed.

Ideally, DOHMH should take part in the OSQA process itself, by participating directly in development to at least some degree. This does not mean that DOHMH needs to be the primary driver of development for any particular component, or even a major contributor, nor does it mean that DOHMH should necessarily be the main supervisory provider of OSQA. It is just that even a small level of direct participation in collaborative technical work will give DOHMH a level of credibility and connection with all the vendors working in the project that cannot be achieved any other way. That credibility will pay dividends in every meeting, progress checkin, and contract negotiation, by alerting external contributors (and especially vendors) that DOHMH understands every aspect of the work. It will also help ensure that open source processes are followed in spirit as well as in letter, since lapses will be more readily apparent to DOHMH.



Key Recommendation: DOHMH should participate directly in technical development, even if only to a small degree, in order to create credibility and connection with vendors and contractors and to contribute to maintaining a consistent open source culture.

Furthermore, by requiring OSQA to sign off on code submissions, DOHMH can enforce standards in the one way that vendors cannot ignore: it stands between vendors and reaching their contracted deliverables. That is, it allows DOHMH to enforce open source process compliance as a condition of being paid on time.



Key Recommendation: DOHMH should require acceptance of code into the open source repository as part of the definition of contractual delivery in software milestones.

At every point, the portal should be ready for open source engagement, and OSQA is DOHMH's assurance that this is true.

Sitting on the tree is only one aspect of an OSQA program. There are a number of other strategies, and choosing the right mix for a given project should occur at an early stage of project specification. Fully describing all these techniques is beyond the scope of this report, but the point here is not that DOHMH should master all these techniques.



Key Recommendation: DOHMH should add OSQA elements to its requirements list when considering project roles. If it does not plan to do OSQA in-house, it should consider contracting for it. For a fuller description of what this element entails, please see a sample OSQA statement of work in Appendix B: “OSQA Example SOW”.

OSQA has a track record of success in New York City. OTS played an OSQA role for the City’s Comptroller’s office during Checkbook NYC development. In that case, OTS was engaged long after the primary development vendor had begun work, and it was too late to fully instill open source practices. Instead, OTS worked with the vendor and the comptroller’s office to identify process failures and teach open source best practices. Over time, vendor participants as well as City staff began to reap the benefits of investing in process. After some time, teams embraced the process because it paid visible dividends (eventually including independent deployment by another jurisdiction, in Texas). OTS was able to back away, leaving the project to manage the process on its own. In later check-ins with the team, we understand that they still rely on these processes as the project continues to develop.

For a project like the portal, OTS recommends adding OSQA capability earlier in the process and combining that role with other project management duties. Putting OSQA at the coordination center of a project allows OSQA to both enforce standards and also shape the process and lead by example. It also puts it in a good position to play a role as advocate for DOHMH in evaluating contractor decisions and proposals.³⁰ Ultimately, learning by example is how well-functioning open source projects set norms, and as the portal development finds its rhythm, one can also expect less-experienced open source vendors to learn from the others.

4.4 Staffing

Staffing a software development effort while engaged in modular contracting can be a challenge. Smaller contracts will tend to lead to smaller vendors as large vendors chase larger opportunities. Different contract modules will have different timeframes, and some vendors might even have discontinuous engagements that end well before the next one begins. This can be difficult for small vendors to manage — they cannot afford to carry idle teammates and will either reassign them to other projects or, in the case of contractors, let them move on to other companies. When it comes time for those vendors to reapply themselves to DOHMH’s portal, they have to source new developers, onboard them, and hope they quickly come up to speed. That process cannot even begin until a contract is signed, and that usually means projects are understaffed at the start.

Surely, some might say, these staffing concerns are not DOHMH’s problem. After all, managing this type of staffing variance is exactly why DOHMH hires vendors in the first

³⁰Interviews suggest that having an advocate in this role would have been valuable during the original contracted development of the environmental health data portal circa 2007.

place. This is true, but that doesn't mean DOHMH avoids the costs and risks that come from a vendor's staffing issues. Schedules will still slip, tasks will be rushed to compensate, and costs will increase. Eventually, DOHMH will pay a price.

There are several approaches to problems of this nature. One avenue would be to design the development schedule to minimize discontinuities in work and contracts. Successful modular contracting requires admitting that DOHMH can, on paper, shift some costs and risks down to vendors, but that a successful project will avoid doing this. For vendors that DOHMH hopes will stick around, structure MSAs that make adding development stages faster and easier. Provide for a certain amount of work between stages so that there are no completely dead periods. The project can use those work periods to tackle the technical cleanup and organization tasks that fast-moving development projects never seem to have time to finish. Keep the team engaged with important, default work that is not tied to specific, big-ticket milestones. That way the team will remain employed and available to DOHMH throughout the project.

At some agencies, procurement processes are prone to long delays in finalizing agreements. Approvals get held up. Contracts must be reviewed one last time. Changes in administration can put everything on pause. It is important for agencies to exercise self-awareness about such constraints and to guard against them.

Here, modular contracting can help by enabling DOHMH to specify work in sets that fit under caps that allow contracting with streamlined processes. Other tactics can help too: Begin procurement processes early, especially renewals for additional development stages. Adopt the MSA structure mentioned above, which has the added benefit of enabling vendors to operate under multiple contract modules at once, each with different end dates. Design contracts to run through periods when agencies might be distracted. It would be unreasonable, for example, to expect the officials at the Board of Elections to approve anything the week before an election.

Finally, there is another approach worth mentioning but not recommending. If DOHMH did not have any goals with regard to growing the pool of City contractors to include more small or minority-owned businesses, it might, as some large companies do, impose a rule that the vendor cannot depend on DOHMH for more than a certain percentage of its revenue. This might result in a preference for contractors large enough to absorb the cost of carrying under-utilized developers, or vendors large enough to have other projects with short term needs for those developers to fill. This might work, though OTS has only seen it used in the private sector, not the public sector. In any event, New York City agencies do have goals regarding small business and MWBE awards, so we hesitate to recommend this approach for DOHMH.

Of course, staffing will ebb and flow as work requires and budgets permit. Nothing in this section should suggest that DOHMH should constantly carry a full complement of developers with no regard to such considerations. The approaches described above are designed to provide DOHMH with tools and insights it can use to take care when structuring successive contract modules. The disruption from small gaps can be larger than intended, but longer-term, planned staffing adjustments are always an available tool, even

when issuing modular contracts.

4.5 Budgeting

One of the deliverables of our analysis is to “ballpark the level of effort needed to build the new system and estimate timelines compatible with DOHMH’s budgeting rhythms.” After wide-ranging interviews, we came to understand that different participants have different — albeit overlapping — ambitions for what should be included in the “next generation” of the Environmental Health Data Portal. This is natural, and indeed is a sign of committed engagement on the part of those responsible for the portal, but it does complicate the task of scoping and budgeting the next phase of work. In some ways it might make more sense to pick a number and a time frame, and then prioritize work within those constraints.

Nevertheless, we came away from our interviews with *some* rough sense of the department’s shared conception of the next-generation portal. It would be based on a new, open source technology stack, it would engage internally and externally using open source collaboration methods, it would expand the possibilities for visualization and data analysis for users, and it would improve the data creation and import process for maintainers. The next phase of development would not seek to replace the existing portal all at once, but rather to replace it in stages, with new components existing alongside old components.

Given these requirements, and an assumed time frame of roughly one year, we will *very* tentatively attempt to ballpark costs. We emphasize that the numbers provided here are fuzzy estimates at best. At this stage, they are useful mainly for general guidance, but should not be relied on for either forecasting or scoping. OTS would be happy to work with DOHMH on more formal scoping process leading to a firm budget and RFI/RFP language; given the information already gathered, we believe this could be done fairly quickly.

We assume that one FTE back-end and one FTE front-end developer would need to work on the portal during that year, that independent OSQA would be present (not FTE, but perhaps around quarter-time), and that beyond those there would be a certain amount of staff time — around half FTE in total — needed for management overhead and public communications. We further assume some extra contracting overhead due to modular contracting (risk reduction comes at a cost, as Section 4.1 discusses), around an extra 10%-15%.

In total, this puts the estimated cost for the initial development push lasting one year at roughly between \$400,000 and \$600,000 — speaking, we emphasize again, very tentatively: it could well be that a closer examination of the next phase’s scope or schedule requirements would change this range significantly, either by changing the total amount or by changing when different stages happen.

This does not suggest that DOHMH should complete this push in one year. Rather, it might well make sense to spread development over a longer term. This would come at a cost. Finishing faster would raise costs, but stretching development out also raises fixed costs that come over time. In this case, DOHMH would begin with a base system, deploy it

alongside the legacy site, and add modular functionality at its own pace until it is time to completely transition to the new system.

Note that one benefit of a modular contracting structure is the ability to ramp development effort up and down in response to budgeting rhythms. Despite the concerns raised above about continuity in contracting,³¹ it is common for government software projects to proceed in fits and starts. The planned growth pattern of a project might involve constant expansion as the effort validates assumptions and begins to return value, but real-world constraints often prevent such ideal scenarios. Budgets are subject to political constraints, shifting priorities of new administrations, and economic cycles. Though moving slowly makes the overall effort more costly and raises the risk of failure,³² it also allows an agency to make progress opportunistically. That ability is sometimes crucial to a project, although in some cases it might also be a sign that the project does not have sufficient buy-in to succeed.

4.5.1 Maintenance and Operations

One aspect of budgeting that is often overlooked at the development stage is ongoing maintenance and operations. A well-designed software package should yield M&O costs that are predictable, and that in most cases are fairly low relative to the cost of development.³³ While there are many choices to be made about infrastructure, resource usage, deployment environments and so on, most of those choices result in fairly similar ongoing cost structures.

Usually, the constraints associated with M&O are related to integrating a deployment process with existing infrastructure. This is an efficiency measure, but also a cost factor. That is, if an organization (or a government like New York City) has standardized infrastructure and process for managing ongoing deployments, the most cost effective approach is to put new technology within that structure. Backups, upgrades, security alerts, uptime management and everything else are handled as a matter of course along with the rest of the technology. Doing any of these in an ad-hoc manner will usually raise M&O costs.

In the case of DOHMH, it has recourse to New York’s Department of Information Technology & Telecommunications (DoITT), which maintains centrally managed deployment infrastructure. Central management, though, comes with its own constraints. OTS research suggests that, like in many cities, the only way to effectively manage an entire city’s worth of technology in one central place is to force all that technology to use the same technology stacks and to deploy in uniform configurations. This allows DoITT to fit deploys into standard processes, automated tools, and low-knowledge management

³¹See Section 4.4 “Staffing”, above.

³²In particular, we have noticed that projects that lack momentum tend to risk cancellation before completion as conditions change. Moving fast reduces the number of shifts a project must weather and also returns value faster, which helps justify pushing through to the end of the development cycle.

³³There may be exceptions to this — projects in which the monthly M&O cost is high relative to the original cost of development — but we do not anticipate this project being such an exception.

structures. This is an effective way to lower M&O costs, and a number of cities have adopted this approach.

Of course, there is a cost to restricting agencies to a set of uniform technologies arranged in pre-approved configurations. Central agencies who have to consider the implications of technology changes City-wide sometimes find it difficult to explore new avenues as technology evolves. It is difficult to approve solutions that are well-suited to the needs of just one deployment scenario but require deviation from the uniform deployment standards. Every such deployment undercuts part of the rationale for centralization.

In DOHMH's case, it is unclear the extent to which DoITT will be able to support deployment of a revised portal within its standard procedures, which makes M&O difficult to predict. If DoITT is not designed to host continuously evolving public projects that make frequent and quick use of open source dependencies, the cost of maintaining an active deployment and performing maintenance-oriented development will fall on DOHMH, regardless of whether that work is done in-house or, as OTS recommends, handed off to an external vendor.

Regardless of where the deployment eventually resides, though, it is reasonable to build a monthly carrying cost into the budget. Even if DoITT hosts the deploy, it is possible (perhaps even likely) that DOHMH will have to be a motive force behind non-feature upgrades for stability and security.

Custom software should not be treated like a depreciable asset that gets fixed when broken and replaced when outdated. Modular design and contracting approaches enable DOHMH to make continuous, small-scale investments in the portal that can extend the lifespan of the portal indefinitely. Even major upgrades and transformations can be performed in stages. The same software development process that performs ongoing updates should also tend to maintenance of those modules. For modules that are not currently undergoing active development, combine the maintenance with slowly tending to structure and paying down the "technical debt" that accumulates in projects and degrades them over time.



Key Recommendation: While operation might be a separate line-item, much of the maintenance work should be part of the ongoing, modular improvement that keeps software current and prevents DOHMH from having to start over.

For a project of the portal's expected size and complexity, OTS recommends budgeting a minimum of \$3,000 per month for software maintenance spread over all the modules as well as \$1500 for deployment and operations costs.³⁴ This amount would cover responding to minor security events, upgrading dependencies, managing backups, systems and database maintenance, and similar tasks. It would not add features, fix major bugs, or address major upgrades (e.g., an event like a major dependency reaching end-of-life).

³⁴This assumes the portal remains a relatively low-traffic site. If it suddenly begins attracting large numbers of visitors, operations complexity and cost would likely increase.

These are “life-support” prices. They don’t buy much, and OTS recommends the software maintenance budget be folded into a modular contract with a vendor whose remit includes a variety of code caretaking tasks that address the types of preventative maintenance and non-feature tasks that keep software ready for improvement over time. DOHMH might budget \$5,000 per month for that role. Because this is a role that engages with many different modules and overlaps with OSQA (ideally, one of the aspects tended to in this role is open source quality), this is a role well-suited for a vendor that also does OSQA and project management.

4.6 Open Source Solicitation

The portal project will proceed using small M/WBE solicitations of less than \$150,000. At some point, though, DOHMH may want to embark on a larger project that does not easily fit into a “small contracting” budgeting provision. In that case, it will need to operate an RFP process. This section contains information on open source concerns that should be considered during that process.

Another aspect of procurement that government agencies might improve is in the solicitation process. A common pitfall is beginning an RFI and RFP process with high hopes for choosing vendors eager to provide agile, open source development in a public spirit of building multi-jurisdictional software. Unfortunately, RFP responses often include a long list of traditional vendors who are not eager to work in this manner and perhaps lack the experience needed to do so well. The hoped-for vendors are nowhere to be found. Eventually, as RFP deadlines loom, agencies look around for additional submissions and maybe even try to promote the RFP in new forums. When FOSS-experienced vendors notice the RFP, they have a short week to submit a hastily compiled bid that shows both their inexperience and the rushed timeframe.

This failure pattern appears at all levels of government and in many different types of agencies. The truth is that even as governments are still gaining sophistication with open source, the commercial FOSS world is also still gaining maturity in navigating government procurement systems. Many open source development companies, especially smaller and less traditional ones, do not have procedures that let them discover open source opportunities at early stages. If government is to succeed at open source, it needs to expand the pool of RFP respondents.



Key Recommendation: It is more important to attract bids from vendors who are experienced at open source development than to attract vendors who are experienced at government contracting.

There are several strategies procurement agencies use in this regard:

First, it pays to ensure that solicitations are promoted in media aimed at open source developers, not just at government software vendors. DOHMH might maintain a list of

community connectors who can promote a solicitation to a wider open source audience. It is important to conduct this outreach early in the process, because newer vendors will need more time than government-experienced vendors to prepare responses.³⁵



Key Recommendation: Conduct FOSS-specific outreach early in the RFP lifecycle.

Second, it is important to be clear in describing project requirements and emphasizing the need for open source deliverables, process, and experience. The phrase “open source” applies in contexts other than software (e.g., open source intelligence), so be sure to spell out the full phrase “open source software” for vendors searching the web or databases for opportunities. The types of vendors DOHMH hopes to attract will be sophisticated about the differences between open source software, open data, and agile development. Be specific and strategic in using these terms, and back them up with questions designed to elicit experience in these domains.

Third, if using modular contracting, make sure vendors understand the breadth of solicitations so they can understand both their specific bid and the overall process.

Fourth, open source vendors expect agile, iterative development. A traditional RFP process often asks vendors to envision the entire engagement and price it as a whole, which requires a degree of pre-planning that open source modular contracting is specifically designed to avoid. Craft a process that is clear about goals and requirements but leaves room for vendors to meet them in flexible ways that might change over the course of the project. This requires being clear about which requirements are truly fixed and which ones were added because they seem likely to be needed on the path toward a complete solution.




Key Recommendation: Although contract amendments are possible (especially if they are just budget reallocation), DOHMH should seek contract terms that allow flexibility and iteration wholly within the terms of the agreement.

The upside for DOHMH is that this flexibility runs in both directions. DOHMH should be able to request incremental improvements that were not specified in detail at project inception without incurring the costs of change orders.³⁶

³⁵Although it goes without saying, it is worth emphasizing here that this type of targeted outreach must comply with ethics and procurement rules.

³⁶This is a concern that arose specifically during the original work on the environmental health data portal circa 2007. DOHMH should avoid vendors that demand high-overhead change orders for minor adjustments. Neither vendors nor DOHMH can be expected to predict every last detail of development in advance. A process that requires such prediction is a broken process.


 **Key Recommendation:** Include in contracts a process (and budget) for iterative process and lightweight changes that do not require giving up other features and milestones.

In talking to open source software development vendors, OTS meets many capable firms that would provide excellent service to government agencies. Too many of these firms avoid applying because they cannot navigate the process. DOHMH can procure from experienced open source vendors by leading more of these vendors into the government services space and fostering competition in FOSS service delivery.

5 Sketching A New Portal

At a high level, requirements for a new portal do not differ radically from the current system. The current portal serves its various audiences well, and everybody at DOHMH manages to use the current set of tools to produce a final product that informs New Yorkers.

However, the process by which DOHMH produces that final product is often inefficient, manual, and ad-hoc, especially when it comes to turning incoming data into published, exportable artifacts or creating new visualizations. Everything from editing workflow and approvals endures a process that has accreted around existing tools but is not well-supported by those tools. The team has adapted to its technology instead of the other way around.

 **Key Recommendation:** The portal should begin with improvements to content, but eventually include in support for workflow around editing and approvals in its plans.

There are other opportunities for improvement as well. The current site has little support for automated testing. Something as simple as a bot that crawls the site, tests for links, and files issues in the tracker would be a big step toward supporting consistently high quality and conveying that quality to the public.³⁷

Similarly, needs are changing. Datasets are growing both in length, but also in width. As the whole world orients itself around more data sources and more data uses, datasets are becoming increasingly detailed, with more columns and complex relationships. Audience expectations are also growing. People that were once impressed with simple visualizations (a map, for example), now want to be able to work with data in more sophisticated ways.

³⁷An open source portal might even open a ticket describing such a bot in hopes it might make a good first task for a project newcomer. It is always useful to have available a set of such “low hanging fruit” issues. They allow new developers to contribute without detailed project knowledge while also starting to provide the missing context that allows deeper engagement.

They want manipulable overlays, multiple maps to allow comparison, and multivariate visualizations. The portal will need widgets that display data in more complex ways and offer users new modes of interaction, and that's even before DOHMH considers mobile users with their small screens and lack of keyboards.

Based on the internal and external interviewees we talked to, the most important features the portal should seek to develop in the near- to medium-term future are:

- Provide continuous access to raw data, possibly versioned, via standard (e.g., REST or GraphQL) APIs and announced via RSS feed.
- Better support for mobile, starting with more use of responsive design.
- Automated testing of links on the portal, with errors filed as tickets in an issue tracker.
- Lower loading time and response time.
- Export of data visualization views that people can put in presentations, reports, and articles.
- Internationalization and localization.
- Tooling and support for translation.
- Accessibility built in from the start.
- Revamp analytics so it is more susceptible to yielding insights about visitor behavior and needs while still respecting user privacy.
- Jupyter Notebook integration, while technically just a specific case of visualization embedding, is an important avenue for enabling researchers to present their data to the public.
- Widgets should update with new data as it becomes available.
- Widgets should have testing frameworks.
- Widgets should report bugs when they crash.
- Increased channels for direct public engagement that allow following up with visitors who want to know more or to discuss the data.
- Build A/B testing as an observability feature of the deployment architecture.
- Improved ability to combine datasets available on the site and view them in different widgets

5.1 Audiences

The portal is notable in that it seeks to simultaneously serve many different audiences, each with distinct needs. Here are just a few of the audiences OTS learned about during its research:

- Researchers are interested in understanding the City and also in collecting City data to compare to other jurisdictions.
- Journalists are of particular interest. They can relate stories important to City life, and DOHMH's data can be the basis of articles about policy, health, and justice.

Journalists can amplify the effect of DOHMH’s work, even if it can be difficult to target them with outreach.

- City staffers use the portal to inform policy and also as public, authoritative sources of information.
- Students in health-related fields use the portal as part of their learning.
- Community organizations rely on the portal to find data that supports their local advocacy.
- Individual citizens, too, come to the portal for information about how the environment impacts their health. Another important constituency is other City agencies that look to DOHMH to inform policy-making.

An effective portal serves all these audiences, giving each what they need in different ways. At the simplest level, research analysts often want raw data, a detailed, sophisticated understanding of how that data was collected, and any conclusions the City has drawn from that data. Journalists want context that helps them tell a story. Citizens often need to understand environmental health impact at a micro level. Peer agencies need DOHMH to explain the data’s meaning in ways that suggest paths toward improvement. The same portal needs to present information suitable for several audiences, each with their own needs, and each with their own level of sophistication when it comes to environmental and data science.

OTS briefly considered a recommendation to break the portal into more focused mini-portals, each aimed at a different audience. We envisioned multiple different front pages that would most easily steer each different kind of visitor quickly to the types of articles and data best adapted to their need and facility with data. Once we began grouping distinct constituencies into different portal approaches, though, this path quickly became unwieldy.

After careful consideration, and discussing the notion with City staffers, we concluded that the additional cost and complexity of maintaining multiple entrance pages was not worth the gains in simplifying the interface for each particular audience. Still, managing distinct groups within a diverse audience comes with its own complexity. Fortunately, there are standard industry practices to help with such situations.

OTS observed different interviewees taking on internal advocate roles for various audience constituencies. This is common in open source work, and is a good sign that portal staff are focused on the practical needs of real users. However, it is also an ad-hoc and potentially failure-prone way to ensure a diverse set of user needs are met.

In situations like this, OTS recommends developing robust user personas as a way to sharpen understanding of users and also to provide a systematic way to include all of them at every stage of planning. It was unclear from interviews the degree to which DOHMH uses formal persona modeling. We received varying answers to questions about who uses the site and why. The site aims at a large variety of users (“public health professionals,

community-based organizations, community boards, City agencies, elected officials, health workers, advocates, and everyday New Yorkers”)³⁸ and one potential cost of that broad approach might sometimes be a lack of crispness in viewing the different portal constituencies. If formal models exist, we recommend applying them to design and analysis more broadly than we currently observe. If they do not exist as fully specified personas, this would be a good time to develop them.

The Internet provides a multitude of resources on developing and improving user personas. Of particular use to persona modeling in the governmental context, <https://usability.gov/> has an excellent guide. In addition, Matthew Montesano has expertise on this topic and is already engaged in user research through the EPHT Portal User Group.



Further Reading: The US Department of Health and Human Services maintains many helpful resources at <https://usability.gov/>. <https://www.usability.gov/how-to-and-tools/methods/personas.html> might be of particular interest to DOHMH.

Once user personas are complete, DOHMH can validate the population of users represented in the Portal User Group. This might lead to more outreach. The personas might also power new user stories that can guide further work. In particular, we emphasize that user stories are useful throughout the project, not just to designers during design phases.



Key Recommendation: Make more use of personas and user stories for key audience segments to help ensure that design and content serve all constituencies well.

5.2 Architecture

Modularity is what enables a site to steadily evolve over time, by upgrading individual pieces of a larger system rather than trying to replace the entire system at once. Every major service on the web is constructed from modular pieces for exactly this reason.



Further Reading: For a primer on 10 common ways software is broken into modules, see <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>.

One of the next big challenges for the portal will be serving mobile browsers as well as it serves traditional browsers. While responsive design techniques will help, it is possible mobile users will need more substantial adjustments for some types of visualization widgets. A modular architecture with well-defined API boundaries might be helpful in developing more flexible front ends that can adapt more nimbly to more circumstances.

³⁸<http://a816-dohbresp.nyc.gov/IndicatorPublic/LearnMore.aspx>

For example, widgets should be modular pieces, and should themselves be composed of modular pieces. It should be easy to add scatter plot functionality to any existing widget on the site. This is not possible if widgets are ad-hoc piles of code developed outside the community process. Or for another example, generalized Jupyter integration throughout the site is an important eventual goal for a new portal.

Jupyter is quickly becoming a standard tool that allows a site to deliver code, data, and dynamic visualizations embedded in a web page. Visitors can be given ways to adjust the code, organize or filter the data, and interact with the visualization. Jupyter excels with data that changes over time. That is, it can load the latest versions of datasets for display and interaction.



Further Reading: For an example and a tutorial on using Jupyter with time series data, see <https://www.influxdata.com/blog/streaming-time-series-with-jupyter-and-influxdb>.

5.2.1 Choosing An Appropriate Platform

It is important to choose an appropriate platform for the portal. There are a range of factors to consider, and DOHMH enjoys a wide range of options. Each has its own set of advantages and difficulties, and while there is no obviously correct answer, OTS recommends a Bootstrap 4 front end supported by a Django back end as a likely best option for the portal's circumstances.



Key Recommendation: A Django-based back end platform coupled with a Bootstrap-based front end would be a flexible and maintainable choice for the next generation of the portal, and would be compatible with the DoITT City Core Framework (CCF).

The platform decision is really multiple platform decisions. Properly designed, the front end is a separate modular piece from the back end. The two communicate over documented APIs and either should be able to be swapped out without overly affecting the other. While those decisions are related, they are not interdependent in any absolute sense. DOHMH can to a large degree choose freely on both sides.

This raises the possibility of using some other architecture, like a static site generator and the JAMstack (Javascript, APIs, and Markup). In OTS's experience, this can be a difficult stack to do well while delivering a complex site. The JAMstack puts a lot of emphasis on API design. It is not easy to create a coherent and efficient API that avoids using too many connections and only passes just the data it needs in each transaction. For a medium-traffic site like the portal, assembling pages on the server is a strategy that can

provide sufficient performance. Still, if DOHMH is using the JAMstack for other projects or if it is considering a hybrid approach, it might be useful to consider it for the portal.



Further Reading: See <https://www.netlify.com/pdf/oreilly-modern-web-development-on-the-jamstack.pdf> for a comprehensive guide to this architecture.

In considering platforms, OTS prioritized the practical need to staff a project flexibly over time, to add new vendors in a modular contracting arrangement (see Section 4.1), and the potential for open source growth of the portal. We considered performance, but at the portal's scale this wasn't a primary differentiating factor, with the possible exception that complex visualizations might struggle on lower-end mobile devices. Those kinds of performance issues, though, are usually best addressed in the visualizations not in the platforms serving them.

5.2.2 Front End Platform

There are four dominant front-end web frameworks: Vue, Bootstrap, React, and Angular. While their functionalities are not exactly the same, they overlap considerably, and each system has its adherents. It is not clear that any is universally superior to the others. For DOHMH's situation, OTS recommends Bootstrap, but it is impossible to say that there is a wrong (or clearly right) decision.

Our recommendation of Bootstrap, tentative as it may be, is based on a combination of factors:

- The technical capabilities of the framework itself.
- Developers' ease in learning it.
- Wide availability of online resources and examples.
- NYC DoITT's choice of Bootstrap 4 for the CCF.³⁹ The potential for interagency cooperation to improve the CCF might be quite valuable, assuming DoITT's code suits the portal's needs.
- Hiring advantage: many applicants will be already familiar with Bootstrap.
- Third-party vendors' familiarity with Bootstrap and ability to support it.
- Independence from a single major corporate sponsor. Although Bootstrap was started at Twitter, its core development team is now spread among other companies (GitHub being a major one, but not the only one).

³⁹See <https://www1.nyc.gov/assets/doitt/html/nyc-core-framework/index.html> and <https://github.com/CityOfNewYork/nyc-core-framework>.

Here is some comparison with the other top choices, React, Angular and Vue:

React has many of the same advantages as Bootstrap, even though their functionality is not precisely the same. React is quite a bit newer than Bootstrap, but is already quite popular and widely used, and has been heavily invested in by Facebook, where it originated.

High-quality examples and learning resources for React developers are easily found on the Internet, and many web developers already have experience using it. While Bootstrap is primarily a front-end framework that provides a consistent UI, good mobile support, and responsiveness to user interaction, React provides a slightly more disciplined development approach that focuses on rendering the DOM⁴⁰ through a rich set of Javascript functions that perform well when handling data that rapidly updates in real time.

Because React checks virtually all of the other boxes, aside from CCF compatibility, it would be a good choice if additional functionality not available in Bootstrap needs to be brought in.

Angular is a complex framework originally created by Google in 2010. It is heavily typed,⁴¹ uses a “Model-View-Controller” style of programming, and boasts a steep learning curve that rewards those who endure the pain of climbing it. Angular is for software development professionals who can make a major commitment to learning a platform that pays off over years of building Angular sites. This makes it a questionable fit for DOHMH’s use. Portal development that requires high up front investment in mastery and expertise will exclude participation by DOHMH’s non-developer staff and present a barrier to “drive-by” contributions from DOHMH and City staffers who are not deeply involved in portal work. A simpler framework would allow DOHMH staff to better understand the technology and participate more fully from the prototyping stage through to live deploy. Every time an internal staffer has to weigh trade-offs in reviewing a development plan, Angular’s cost will hurt the project. If DOHMH wants to involve a wide range of developers in the project in the way described during OTS’s research, we recommend looking at other options.

Vue is somewhat more recent (2014) and still has a smaller following than the other three, although it is gaining in popularity. While it lacks the backing of React’s Facebook or Angular’s Google, this also means it shares with Bootstrap the advantage of not being subject to the changing business needs of any one company. If either Facebook or Google were to shift to new frameworks, it is unclear how their existing frameworks would survive, as those companies provide a large portion of their development resources. Vue’s long-term prospects look good and its independence from any single sponsor is a solid point in its favor.

In the end, though, the advantages of Bootstrap, especially CCF compatibility, tip us toward it as the first choice. Bootstrap resources, expertise, templates, extensions, and examples abound. This flattens the learning curve for Bootstrap in ways Vue cannot currently match. Although Vue is known for its simplicity, its learning curve is less

⁴⁰The “Document Object Model”, that is, the data structure that is rendered to become the web page that the user sees.

⁴¹Here, “typed” refers to a technical property of the framework, a property that generally leads to more reliable code but at the cost of greater up-front discipline on the part of developers.

supported than Bootstrap’s. That, coupled with Bootstrap’s flexibility — it can be used like a toolkit and integrated with other systems, including with other front-end frameworks like React — makes it a tool that would serve DOHMH well no matter how the portal’s needs change over time.

5.2.3 Back End Platform

There are a variety of back end web platforms that can support a site as unique as the portal. The two most suitable options are a site that starts by customizing Drupal or one that starts with Django. There are different trade offs for each option, and in the end OTS recommends Django for this particular application.

Drupal is a PHP-based CMS with a large following in civic tech circles. PHP is a widely used language, which means PHP developers are relatively easy to find in the labor market. However, PHP is a programming language that takes experience to write well, which means that in practice DOHMH would likely be hiring from a subset of the wider pool of developers.⁴² Still, it is likely the case that DOHMH would be able to staff a PHP-based project without difficulty.

Django is a Python-based web framework. It is not a content management system at all, but is instead a base of infrastructure and support for building web applications. Python is a well-structured language that likes to present itself as “batteries included”. Its ecosystem boasts an enormous range of modules and recommended patterns that can provide functionality to meet just about any need. The same cannot be said of PHP.

Because Python is a relatively easy language to learn and its modules allow it to operate in any domain, it has emerged as one of the two main languages of the data science world (the other is R, a language that focuses on statistics computation). Given the portal’s mission, this is a large advantage for Django. Many of the people who engage with the portal for substantive reasons will arrive with skills that would allow them to engage in Django development, and DOHMH already has in-house staff with Python experience. Choosing Django would give DOHMH a participation pipeline for visitors and, more importantly, increase the potential for the portal to see adoption in other data communities, including environmental health communities.

Django is built on a set of tools designed to work with models of data objects. That also makes it a good fit for some of the types of things DOHMH might do with the portal over time. Django also has a fairly well-developed ecosystem of packages⁴³ to draw on.

Note that Drupal is not without its advantages too. It comes with much of the editing and administration workflow built-in. There are a wealth of plugins to change behavior and DOHMH can do custom development to suit that workflow to its needs. Administrative

⁴²Furthermore, there are also many examples of bad PHP on the Internet, and there are not always clear signs by which to distinguish bad examples from good ones. Thus, if one is not already good at PHP it is, relatively speaking, somewhat harder to get better at than other programming languages are.

⁴³What we might normally call a “plugin ecosystem” is referred to in Django terminology as a “package” or “module” rather than “plugin”.

interface development on a Django site is likely to be 20% to 25% of the software development budget at the start of the project.

OTS balances Drupal's benefits, though, against the main cost of adopting it: customized versions of Drupal are pegged to a specific Drupal release. As Drupal rolls over to a new major version, upgrading is an unpredictably large effort. Customizations will need to be adapted; plugins will break; themes will need adjustment. Keeping up with Drupal through even one of these transitions is a major cost, one that is hard to justify since it does not add new features to the site. Many projects simply stop upgrading at that point and begin planning their next major rewrite. This is a major barrier to the continuous, modular development process OTS recommends throughout this document. This is especially true for government projects that often see service for longer-than-anticipated periods of time before they can be replaced.

There are other considerations in choosing between these two platforms, but at the highest level, these are the ones that appeared most dispositive. Other platform options include Joomla, WordPress, and some of the node-based CMS platforms. In examining each of these, OTS concluded that none of them presents any significant advantages over Django or Drupal, and the ecosystem advantages of choosing one of those two options outweighed other considerations.

5.2.4 Maintaining the Legacy Portal

Modular development of a new portal will require maintenance of the old portal for some period of time. OTS recommends opportunistic replacement of old portal content as the needs, opportunity, and funding for upgrading legacy content present themselves. This will require a technical infrastructure design that can readily mix content from multiple systems.

There are several approaches DOHMH might take to integrating old and new portal content. The simplest is to operate both portals at once and link between them as needed. This strategy is easy and keeps the old portal from interfering much with the new portal's design. It will not, however, present a coherent whole to end users. The two portals will have different interfaces and design language during the transition period, which could last a long time.

Another approach is to schedule migration toward a new portal in stages. That can be done if the new portal is feature complete in discrete sections that can take over for the old site.

As the portal repository grows to include new widgets, DOHMH might find that it wishes to include some of these in sections of the site that are still tied to the legacy portal system. This might be an indication it is time to port that section of the site to the new system. It might also indicate a need for effective, modular widgets that can be easily injected into the legacy pages. There are tradeoffs between these two approaches, and the development team will need to consider them in formulating a policy for the project and a plan for specific instances.

Finally, DOHMH might enable the new portal to present aspects and parts of the old portal, and to sunset this practice over time as new functionality and content are ready. It might then preserve the old portal at its old URLs indefinitely or simply forward those URLs to pages of the new portal that most match the old one’s topic.

At some point, the cost of maintaining two systems, even if one is not under active development, will be a compelling reason to migrate what content remains and to sunset the old portal.

5.3 User Interface Concerns

The data portal contains excellent tools for examining and interacting with data. There are, of course, a range of improvements that could enable it to better serve its community of users.

5.3.1 APIs and Feeds

Research analysts who want access to data might gain that access in a few ways. Currently, some pages have download links⁴⁴ that yield tabular (i.e. Comma-Separated Value (CSV)) files. There is no visible way to query data via a more dynamic REST or GraphQL interface or to get timely updates of new data. Integrating the data portal into dynamic or live systems would be difficult.

Although DOHMH delivers data to the public, much of the data the portal relies on is aggregated from other sources, including other City data portals. That is, DOHMH is not the original source of the data. The portal just happens to be the place where the public finds and downloads it.

Despite this, a future portal should make it easy to turn CSV and spreadsheet datasets into feeds and queryable services. Ideally, this queryable interface serves DOHMH as well as the public. That is, researchers would benefit from a standard way to access DOHMH data, and that is true regardless of whether those researchers are inside or outside DOHMH.

Even further, this data aggregation and publication layer should be a separable open source module with generalized interfaces. Ideally, DOHMH can adapt something that already exists, but what is most important is developing a component that might serve other agencies. If DOHMH can contribute to standardization and interoperability among City agencies, it can improve data access for everybody and enjoy a system that provides queryable data with less manual intervention. The best way to do that is to work with other agencies (inside New York and even beyond the City) to build general-purpose data lakes⁴⁵ that lets different agencies access public data in standard ways but for diverse

⁴⁴Note that during in-house testing, OTS was unable to make the “Export” links work consistently. It’s unclear whether the links failed or if it might work only after a long delay after clicking it. Either way, it did not provide access to exported data within a reasonable timeframe. OTS conducted such testing from different locations, from different brands of browsers with stock settings on two separate occasions.

⁴⁵https://en.wikipedia.org/wiki/Data_lake

purposes. It is possible (perhaps even likely) that other City agencies are already working on this problem, perhaps as a joint effort or in siloed, internal projects. In particular, it would be worth consulting the Comptroller’s office, as Checkbook NYC provides downloadable datasets.



Key Recommendation: Identify other City efforts to manage and distribute datasets and work with sister agencies to deliver this data in standard ways.

The biggest win from standardization is that DOHMH can then improve tools and processes that streamline creation of visualizations or distribution of more complex (e.g., filtered or joined) datasets. Increased tooling and automation of joining data to widgets is a necessary step in shortening the lead time of articles, lowering costs, and providing better data to the City.

5.4 Editing and Admin Workflow

Currently, there is a workable set of steps that turns research into stories and public resources.

One aspect of open design is that nobody needs to draw lines around who has input on new user interfaces. An open source project can publish in its repository user research artifacts (e.g., personas, stories, and surveys) along with proposed designs. Anybody involved in the project is then invited to participate in feedback and testing. This relieves the project from having to decide whom to include and allows participation from a wide range of people, including portal users outside of DOHMH.

This type of open process is typical in FOSS projects in most areas, not just in design. Decisions are normally made in public, inclusive forums, and anybody who participates seriously in the discussion can have at least some voice in the outcome. These sorts of open processes stand in contrast to processes that are overfitted to visible stakeholders. Such processes are a barrier to participation by anybody except those stakeholders, and thus can be a barrier to project growth — especially to growth into unexpected areas.

The portal’s current publishing workflow involves a lot of support and manual processing. Researchers write stories and receive a fair amount of help from the portal’s Data Manager, who supports the final product by, e.g., generating and tweaking maps for publication.

Similarly, there is a lot of manual work involved in cleaning up datasets for publication. After that cleanup, the approval process is insufficiently supported by infrastructure. Internal controls are informal, workflow is unmanaged, and courtesy checks with outside agencies are not tracked or supported by infrastructure.

More broadly, automated interface testing would raise portal quality. As mentioned above, a bot that tests for broken links and creates tickets for them in the issue tracker would be

fairly simple to write (it might even make a good hackathon topic for students). Similarly, it might be useful to test data feeds for availability and comprehensible results. Python scripts that make use of scrapy⁴⁶ are enough to get started.

More automated interface testing would require a more dedicated program. Most such testing is done via Selenium.⁴⁷



Further Reading: For a comprehensive introduction to automated user interface testing with Selenium, see <https://www.guru99.com/introduction-to-selenium.html>.

5.4.1 UX Testing

There is one key recommendation OTS often makes with regard to developing new interfaces: it is important that the portal serve real rather than presumed needs. The only way to achieve that is to spend the effort needed to do rigorous usability testing. In DOHMH's case, there is design expertise in-house, Matthew Montesano, who is already engaged in specifying and performing such testing.

Testing starts with users who come to the portal with a practical need for information. DOHMH knows this, and Mr. Montesano is engaged in outreach and interviews with such users, complete with mock-ups and testing scripts. That work appears to be outwardly focused, and it was unclear to OTS whether there are additional opportunities for user feedback from within DOHMH and related agencies. It might be valuable to review efforts to systematically collect feedback from those users because they are readily available and perhaps form a different constituency than the current testing pool.


Given the excellent work underway, our recommendation for improvement in this regard is incremental. In conversation with staff, OTS did not observe a widespread view of usability testing results. That is, Mr. Montesano's research informs the design effort but conclusions from that research might also be valuable across the rest of the project. Testing yields information about the current design iteration, but it should also describe user needs generally and in ways that are wholly separate from the current design. That information should make its way to every participant in this work.

Note that user interfaces need not be complete before testing them with users. Modern web development tools make it possible to quickly implement live prototypes, even if the data hookups are not always fully in place, and DOHMH should take full advantage of this when seeking early feedback from users. Testing is best done with live prototypes that react in real time to user interface gestures, and this is preferable to testing with mere visual mockups. Static mockups of screens, with only narrative paths or simple click paths provided for navigation, convey too little about the actual experience of using an application. OTS recommends that DOHMH might avoid them where possible.

⁴⁶<https://scrapy.org/>

⁴⁷<https://selenium.dev>


Even further, we recommend design testing be informed by the user personas described above (see Section 5.1 “Audiences”) and also communicated to the team in those terms. DOHMH might also make more use of analytics to track existing visitors as they travel through the site. In interviews, it appeared this type of data was not readily available, or at least not readily in use, throughout the project. These approaches to flowing user-centered design throughout the project are meant to extract greater total benefit from the research sources DOHMH already has.

 **Key Recommendation:** Gain a more detailed view of users from: examining analytics, exploring logs of search queries that direct people to the portal, or by implementing an in-site search bar.

OTS identified one area where usability testing might be delayed or reduced. The portal makes good use of data interaction widgets that provide visualization but also allow users to explore data. These types of widgets are likely to evolve quickly. Some will be instances of generally-useful widgets that get reused. Others might be heavily modified or purpose-built for a specific article. Either way, it is common for such widgets to include interface designs that side-step the early stages of the normal user-testing procedure. Such interfaces might still undergo final user acceptance processes before publishing live, of course.

Furthermore, data-interaction is a young field. Interaction patterns that make sense today might age poorly as design language and audience sophistication evolves. Regardless of whether DOHMH applies extensive tests to data interaction widgets before deploying them, it should reconsider their interfaces periodically over time.

DOHMH must at some point decide what type of process to apply to such widgets. In an ideal world, users would never see any aspect of the site that has escaped rigorous testing. In practice, this type of testing takes time and resources that poorly match the timelines and budgets of many individual articles. Striking the right balance is a matter of policy without a clear, objectively correct answer.

 **Key Recommendation:** Consider creating a usability testing processes that can do basic validation of user interface designs for data-interaction widgets while still allowing the development and deployment speed those widgets typically need.

Another aspect of interface testing that could use improvement is regression testing. Because the site lacks robust automated interface testing, it relies on DOHMH staff manually finding bugs, either by testing problem areas or by encountering during other use of the site. This leaves the site broken in small but significant ways. It would be valuable for the new portal to replace or augment this manual interface testing with some amount of support from an automated process.

6 Thanks




This report is the culmination of extended research into the existing portal, its underlying technology, a wealth of documentation, and the generous time that the DOHMH team spent with OTS. Many people were gracious enough to share their knowledge and unvarnished views so that we could make informed recommendations and contribute to the next iteration of the portal. We at OTS hope we have repaid that investment by providing actionable recommendations that DOHMH will be able to use.

Thank you especially to:

- Michael Porter
- Grant Pezeshki
- Matthew Montesano
- Nancy Jeffery
- Wendy McKelvey
- Sarah Johnson
- Carolyn Olson
- Kevin Anderson
- Douglas Kim

And, finally, thank you to BetaNYC and Noel Hidalgo for partnering with us on this project and providing expertise on municipal technology development, open data, and practical community impact.

This document is built on open source tools and resources. It is written using LaTeX⁴⁸ and contains icons from the Noun Project.⁴⁹ In the spirit of open source attribution, we list the icons and their sources below.

Icon	Noun Project ID
	1785639
	3027864
	2591756

⁴⁸<https://www.latex-project.org>

⁴⁹<https://thenounproject.com>

7 Acronyms

This chart lists acronyms used in this document, their expanded meanings, and the page on which they first appear.

API	Application Programming Interface	13
ASF	Apache Software Foundation	12
BESP	Bureau of Environmental Surveillance and Policy	9
CCF	City Core Framework	48
CDC	Center for Disease Control	8
CSV	Comma-Separated Value	53
CLA	Contributor License Agreement	12
DOHMH	Department of Health and Mental Hygiene	3
DoITT	Department of Information Technology & Telecommunications	40
EF	Eclipse Foundation	12
FinOS	Fintech Open Source Foundation	12
FOSS	Free and Open Source Software	6
GSA	Government Services Administration	22
LF	Linux Foundation	12
M&O	Maintenance & Operations	5
MSA	Master Services Agreement	32
OSQA	Open Source Quality Assurance	4
OTS	Open Tech Strategies	3

Appendix A: Open Source Analysis Checklist

This appendix contains checklists that support decision-making related to whether an open source strategy fits a project, forming that open source strategy, and implementing it.

The lists contain many considerations and exercises that can aid in making and explaining open source decisions. These lists are neither exhaustive nor mandatory. That is, no project will check every item on every list. And many projects will find additional analysis helpful. These lists are meant to provide a wide-ranging menu of options that can help a project decide its own path toward evaluating and implementing an open source approach.

This checklist is fairly generic and organizations will benefit from tailoring it to their existing policies, workflow, and priorities.

Appendix A.1 How Does This Project Thrive As Open Source?

1. Goal Setting
 - (a) Understand mission goals
 - (b) Clearly define overall project goals
 - (c) Clarify how project goals support mission goals
 - (d) Choose 3 open source goals
 - (e) Clarify how open source goals support project and mission goals
2. Open Source Readiness
 - (a) Evaluate the project team for open source readiness
 - (b) List readiness gaps
 - (c) Identify extra-project resources (either internal or external to DOHMH) to remediate gaps
 - (d) Identify open source project champions among senior leadership
3. Archetypes
 - (a) Choose one more open source archetypes
 - (b) Identify exemplar open source projects
 - (c) Detail similarities and differences with exemplar projects
 - (d) Document and resulting strategic concerns
4. Participants
 - (a) Map the ecosystem of potential participants, partners, contributors, adopters, and projects
 - (b) Identify internal partners for immediate contact
 - (c) Identify external partners for immediate recruitment
 - (d) Identify likely first contributor
 - (e) Identify first adopters (or wave of adopters)
 - (f) Form outreach plans
5. Existing Projects
 - (a) Map any competing or overlapping products and projects
 - (b) Differentiate DOHMH work
 - (c) Identify how open source might allow DOHMH work to find a unique niche among these products and projects

6. Resources

- (a) Map the flow of resources (financial, in-kind, and open source) in and around the project
- (b) Identify donors that are particularly amenable to open source work
- (c) Identify likely donors and funders, with a contact plan for each
- (d) Sketch a sustainability model projected over time for likely scenarios
- (e) Detail the ways open source ensures project impact if donors or DOHMH withdraws

7. Vendors

- (a) Map the various vendors who could service this project's ecosystem
- (b) Clarify how the project will establish a multi-vendor ecosystem

8. Identify target organizational hosts for the work, whether DOHMH, Linux Foundation, Apache Software Foundation or other.

9. Articulate how open source provides resources and incentivizes beneficial behaviors in your ecosystem.

Appendix A.2 Implementing An Open Source Strategy

1. Outreach
 - (a) Effect outreach plans
 - (b) Begin recruiting open source project champions among senior leadership
 - (c) Invite planning contribution from potential participants
2. Project Initiation
 - (a) Choose a project name
 - (b) Create a public code repository where work will be done from day one
 - (c) Configure version control
 - (d) Create a public website for the project
 - (e) Add project mission statement to documentation and website
 - (f) Initiate documentation for project (wiki, readthedocs.org, FAQ, README, etc.)
 - (g) Configure issue tracker
 - (h) Establish communication channels (mailing list, chat, etc)
 - (i) Add developer guidelines to repository
 - (j) Add rough draft prospective roadmap to repository
 - (k) Draft accessibility and internationalization plans to ensure the project's output is accessible to all, without barriers of abilities or language
3. Open Source policies
 - (a) Code of conduct policy
 - (b) Outbound license
 - (c) Inbound licensing policy and contributor license agreement
 - (d) Commit access policy
 - (e) Peer review policy
 - (f) Trademark policy
4. Modular Contracting
 - (a) Staff the OSQA role (internal? vendor?), and have OSQA support modular contracting processes
 - (b) Break any vendor work into modular contracting units
 - (c) Draft a CFP for vendors that increases response from experienced open source vendors
 - (d) Put OSQA between vendors and contract deliverables

5. Continued Planning

- (a) Repeat planning, mapping, goal setting, and readiness exercises annually with the internal project team and the distributed contributor community
- (b) Make an outreach plan for new ecosystem participants, whether adopters, contributors, or donors

Appendix B: OSQA Example SOW

Overview

Following the open-sourcing of Software, Vendor proposes to assist Client State to:

- Monitor and improve deployability: Monitor and advise on Software's deployability by other jurisdictions; improve Software's deployability as needed.
- Improve deployment process: Analyze, improve, and document Software's deployment process.
- Manage third-party contributions: Help manage code contributions and other contributions from third-party contributors.
- Assist with public communications: Help inform governments and the civic technology community about Software's availability and capabilities.
- Plan and advise on long-term maintenance: Help find or create an appropriate long-term home for the Software source code and related assets, to support long-term maintenance while also ensuring that State's needs with respect to Software continue to be met. This involve forming or joining a consortium.
- Develop open source contracting and procurement language: See description below.
- Advise on open source / open technology processes: See description below.
- Hackathons / Events: Organize and/or assist in organizing hackathons and other events to promote actionable interest in Software.

This proposed Statement of Work describes mainly ongoing processes. Although each process will likely have specific deliverables associated with it, not all of those deliverables can be predicted in advance. Therefore, this SOW proposes activities with time boundaries, not with the intention of ending the work when the time limit is reached, but to give State a scheduled checkpoint to assess the quality and direction of the work so far, with the option to extend and/or amend the contract if desired.

Assistance Areas

- Monitor and improve deployability
Vendor will regularly monitor the deployability of the Software code and sample data, both by self-deployments and by maintaining regular contact with other parties attempting to deploy Software in real-world environments. Where possible, Vendor will seek to improve Software's deployability. Maintaining easy deployability is crucial in Software's success, not only in jurisdictions outside State but, in the long run, even for State itself. Ease of deployability ultimately means not only ease of

adoption, but faster turnaround times on bug reports, greater flexibility in live deployment options, and expanded ability for others to participate in release testing.

It is typically somewhat difficult for a primary vendor who controls the live production environment to monitor generic deployability effectively. There are too many pressures on that vendor to focus all effort on deployability for the specific case of the primary client's data in that client's environment – even though in the long run, improved general deployability also results in improved deployability for the production environment too. Vendor thus proposes to serve as an independent third party, ensuring that deployability standards are maintained.

- Improve deployment process

Ease and standardization in Software's deployment process are crucial for Software adoption. The first step for any other jurisdiction interested in Software is to stand up a test instance with their own data; without a reliable, well-documented data-import and deployment process, this first step will continue to be a major hurdle.

Vendor will perform technical analysis and make technical and documentation improvements as needed to ensure that deployment is manageable for a wide range of jurisdictions and vendors.

- Manage third-party contributions

Many aspects of managing incoming open source contributions (core code, documentation, sample data, third-party deployment scripts, bug reports, etc) do not require deep technical expertise in the software. Vendor proposes to handle, as an ongoing process, the parts of contribution management that can be separated from in-depth code review and technical decision-making. State and Vendor would of course still make final decisions about what contributions to accept, and how, but Vendor would take care of the entry-level open source project management bureaucracy as much as possible, including but not necessarily limited to:

- Day-to-day communications with participants in the project's open source forums;
- First-pass review of contributions, to ensure proper formatting, description (log message), etc;
- First-pass review of public bug reports, to have initial dialogue with the reporter, resolve duplicates, make sure proper reproduction recipes are included, answer common questions, spot trends in the user community, etc.
- Monitoring of the public discussion forums, to make sure the right parties are talking to each other, to help gather a formal knowledge base (e.g., FAQ) about the software, and to flag important items for State and/or Vendor's attention
- Management of Contributor License Agreements, to make sure the project can legally and safely accept the contribution;
- Management of contributors in GitHub;

- Management of public-facing documentation, to the extent that deep technical knowledge of Software is not required (which we anticipate being a fairly broad extent);
- Coordinate with Vendor on contributions that are being accepted into the core code for general release, to make sure that the internal development process is plugged into the public forums in ways the community can use.

We believe that this semi-separable part of contribution management is probably more than half of the typical public engagement overhead of running an open source project – speaking roughly, somewhere between 70% and 75% of that overhead. Having Vendor handle this portion would allow Vendor to concentrate on technical review and on just the communications that directly involve core code or State-desired features, and maximize the project’s ability to get the full advantage of public engagement, which, if done right, far outweighs the overhead. Vendor would of course remain free to become as involved with public technical engagement as they wish to be and have the bandwidth to be, and Vendor would fully support them in this.

- Assist with public communications

Running an open source project with a primary stakeholder of State’s size inevitably involves moments when messaging (about the inclusion or exclusion of a core feature, for example) must be done in a careful and well-considered manner, in ways that will be comprehensible by both the open source and civic technology communities – including government officials and other vendors.

There are more and less successful ways to send such messages. The more successful ones leave all stakeholders feeling confident about the project and its future (even if their particular concern or bug was not addressed, or even was exacerbated). The less successful ones leave some stakeholders feeling uncertain, and possibly reconsidering their involvement or their adoption of the software. Vendor proposes to assist with such public communications, taking an advisory role in both drafting and reviewing them, in helping to determine which forums and media to broadcast such messages in, and in managing responses to incoming communications.

- Plan and advise on long-term maintenance

The Software’s long-term maintenance may involve participation by multiple cities and vendors. Vendor proposes to help State determine the best shape for this long-term maintenance structure and, depending on the outcome of that process, help to set it up. This will involve discussions with State as well as with vendors such as [...], open technology consortia and organizations such as [...], and others.

Vendor will provide both structural advice, negotiation help, and, if necessary, legal assistance in setting up or joining such a body, in such a way as to ensure that State’s interests are protected along with the long-term health of the Software code. We emphasize that the success of any such consortium-based maintenance plan depends much more on the technical success and adoption of Software itself than on

the particular arrangement of the consortium: no shared maintenance effort can succeed if the software itself is not successful, while even a flawed shared maintenance arrangement can still succeed quite well if the software is fundamentally healthy and in demand.

- Develop open source contracting and procurement language

Vendor will work with State to collect and analyze (from an open source development perspective) the State's experiences in the Software's procurement and development contract, and suggest language or process improvements that could help build open source practices into Software and future software projects from the start, in a way compatible with State's existing procurement procedures.

- Advise on open source / open technology processes

During the phase leading up to the open sourcing, we discovered that various unforeseen questions related to open source processes and technology tend to arise. Their exact topics and timing at least cannot be reliably predicted (for example, in the couple of weeks immediately prior to this draft SOW, we discussed open source databases and deployment procedures in some depth, security audit contracting & timing, and other things). However, statistically, the overall rate and complexity of the questions is somewhat predictable, so we propose to simply budget for it explicitly in phase two.

- Hackathons / Events

Vendor will assist State in determining what developer-oriented events (such as hackathons) would be most effective in promoting Software usage and adoption, and as appropriate organize or assist in organizing those events, including promotion, management of the event itself, and post-event results processing.

[Describe hackathons and events more fully]

Schedule and costs

(Omitted.)

Position Descriptions

- Open Source Program Manager:

Minimum general experience: Experience as a developer and community manager on multiple open source projects; experience releasing third-party code as open source; experience using and deploying open source software in large-scale enterprises. Has active account on one or more public open source project hosting sites.

Function responsibility in project: Oversee entire engagement. Coordinate technical work (including in-house and third-party technical work) to use open source community involvement as much as possible in meeting client's needs. Assist client with public messaging, with building a maintenance consortium, and with monitoring

and maintaining consistency between different instantiations of the code base. Monitor deployability and maintainability, including organizing and channeling third-party feedback on deployments and on data import/export process. Organize hackathons and events related to the open source product. Advise on open source and open technology process generally, including but not limited to advice on RFI/RFP language and contract language.

Minimum education: [...]

- Open Source Project Manager:

Minimum general experience: Experience as a developer and manager on multiple open source projects. Has active account on one or more public open source project hosting sites.

Function responsibility in project: Manage day-to-day interactions with client, especially on technical work. Manage day-to-day interactions between Software Engineer and client's other contractors who are involved in technical work on the open source product. Help ensure that all contributions, including those from client's other contractors and from third parties, are properly and consistently incorporated into public open source repositories as appropriate. Monitor community forums, bug tracking database, wiki, documentation, etc. Provide prioritization guidance on technical tasks. Manage Open Source Software Engineers.

Minimum education: [...]

- Open Source Software Engineer:

Minimum general experience: Experience as a developer on at least two open source projects. Has active account on one or more open source project hosting sites.

Function responsibility in project: Make technical improvements to the open source product as required for successful deployment by third parties. Code, document, and test various aspects of the product, including but not limited to the data import process, data export and APIs, and deployment scripts. Coordinate with other engineers making technical contributions, including both engineers from client's other contractors and from involved third parties.

Minimum education: [...]

Appendix C: Ecosystem Maps

There are many ways to draw an ecosystem map for an open source project. A typical map shows relationships among at least these elements:

- *Contributors / Developers*
- *Service Providers / Support Providers*
- *Instances / Implementations / Known Deployments*
- *Partner Organizations / Co-Investors / Funders*
- *Users (“End Users”)*

Figure 1 shows a simplified real-life example. It is based on an actual map drawn for the Arches Project (archesproject.org)⁵⁰, although the version shown here is greatly simplified and abridged.

The purpose of an ecosystem map is to help you spot patterns and anticipate the effects of contemplated actions. The methodology is intentionally flexible:

- You could use differently-sized circles to differentiate between larger and smaller organizations (larger or smaller in absolute terms, or in terms of the entity’s degree of involvement in the project).
- You could draw lines of communication or cooperation between different nodes. Those lines could use different styles or colors to represent different types of connections. (In the actual Arches project map such lines exist. We didn’t include them here because it would have cluttered up the example diagram too much.)
- While this ecosystem map happens to show geographic regions, not all ecosystem maps need do so. Other kinds of groupings might be more useful for other purposes.
- The key distinctions (“Contributor”, “Instance/Deployment”, etc) may be different for your project. The ones offered here are just a suggestion.

⁵⁰Some background will help: Arches is an open source platform for managing cultural heritage data, started by two sponsors, the Getty Conservation Institute and the World Monuments Fund. It quickly grew to involve a number of different participants. Some of them are cultural heritage organizations (e.g., Historic England) that both contribute to Arches development and deploy instances of Arches themselves. Others are commercial service providers who deploy Arches on behalf of customers. Still others represent deployments (and thus indirectly those deployments’ users) by influencing the project through feedback in the project’s forums and participation in user workshops, conferences, etc, more than through direct code contribution.

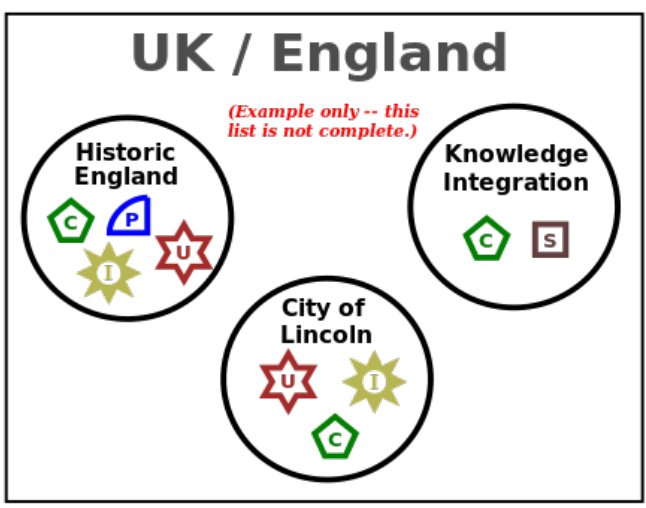
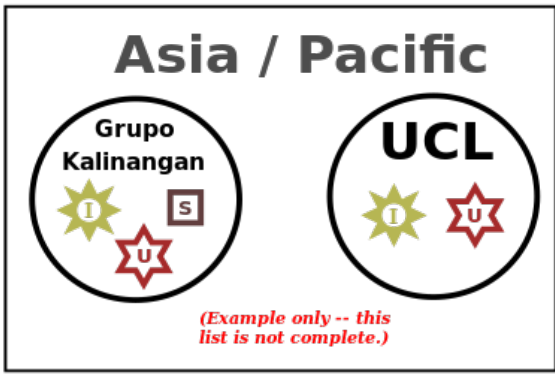
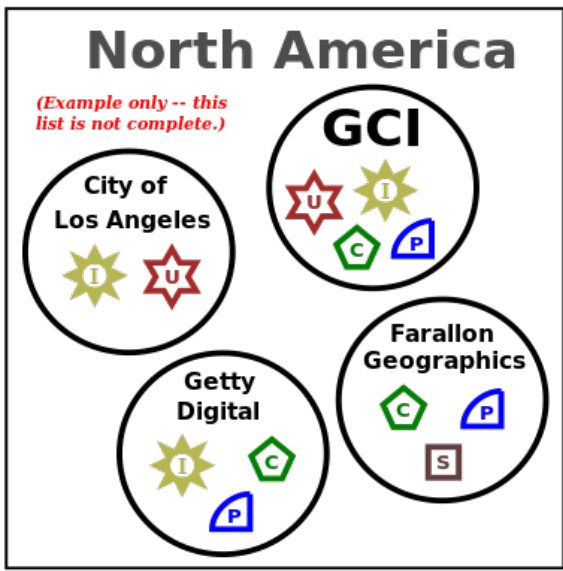


Figure 1: *Example ecosystem map: a simplified and abridged representation of the Arches Project ecosystem.*

Ecosystem maps are meant to be messy, quick, and frequently redone. The best way to make one is to hand-draw it on a large piece of paper or on a whiteboard, and ideally to do this as a group exercise. Figure 1 shows only a few entities, for the sake of fitting the example on this small page; a typical ecosystem map would be larger and have many more entities.

Sketch out the diagram first, being as inclusive as possible, and then look for patterns. For example, if you notice that entities of a particular type — say, small companies, or entities located in a particular geographic region — offer service and support but are *not* contributors, that raises the question of whether the project could do more to help them become contributors. If those entities share a linguistic region, maybe the project should invest in translating its documentation into that language.⁵¹

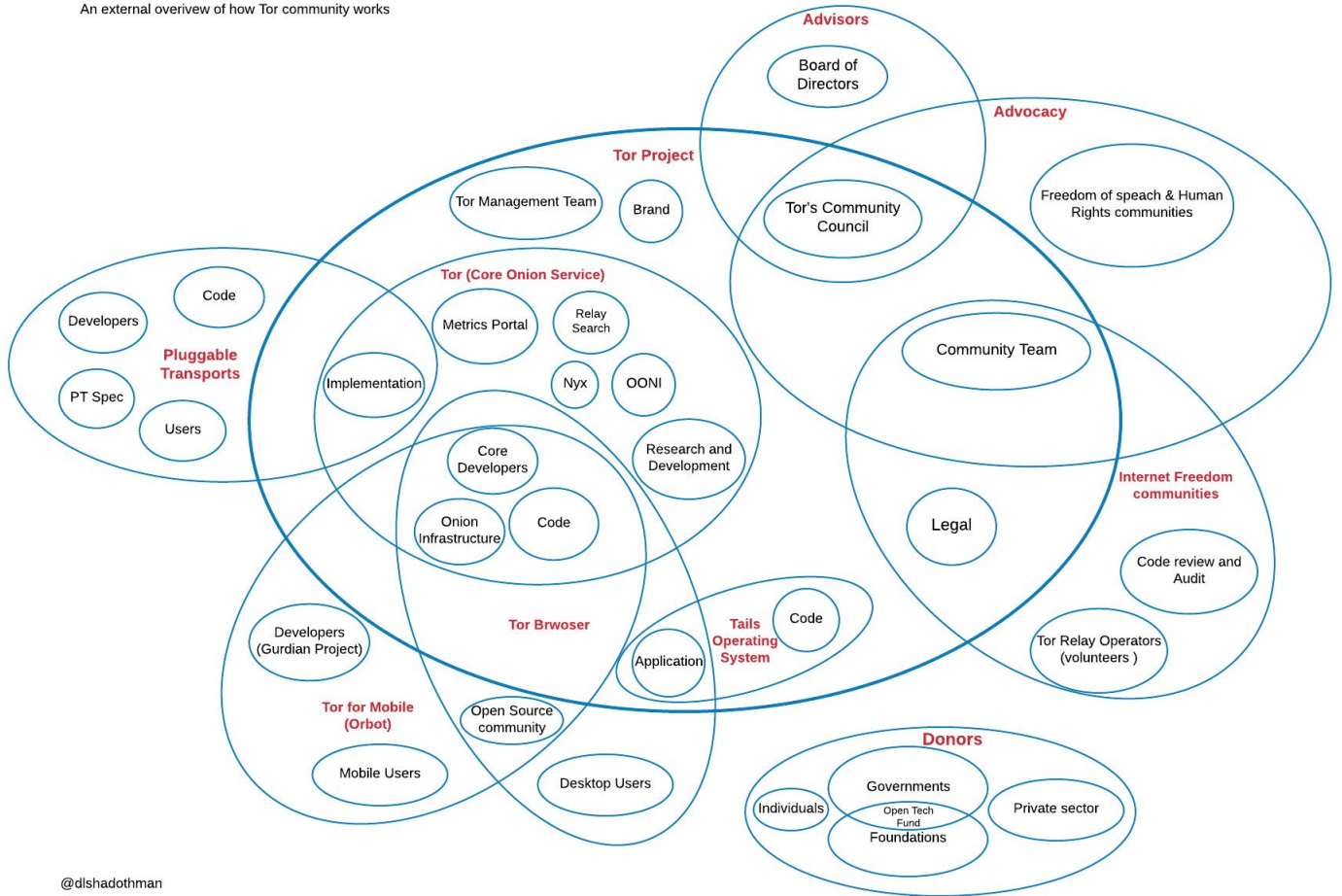
⁵¹It would also be a good idea to look to see if those entities have already created their own language-specific forum in which they discuss the open source project — in other words, your open source project

Ecosystem maps should be drawn for specific purposes. For example, the map in Figure 1 was mainly drawn to help understand *who* the participants in that ecosystem are and what their motivations are. On the other hand, the map shown in Figure 2 was drawn to help understand the flow of information within the Tor project. It does not list external collaborators or deployments individually, nor show geographical regions. Its primary value lies in giving collaborators an overview of all the interest groups around the Tor project and how they relate to one another, by showing which ones are “nearer” and “farther” from each other in terms of communications and concerns. I.e., it’s a quick way to help someone answer the question “Have I thought of every type of participant who might care about what I’m proposing?”

may have grown without your knowing it yet! In such cases, it is often worth investing in some bilingual developers or translators to help bring the two groups closer together. The report “OpenDRI and GeoNode: A Case Study On Institutional Investments In Open Source” (<https://opendri.org/wp-content/uploads/2017/03/OpenDRI-and-GeoNode-a-Case-Study-on-Institutional-Investments-in-Open-Source.pdf>) talks more about the importance of inter-lingual connection in open source (p. 37).

Tor Project System MAP

An external overview of how Tor community works



@dlshadothman

Figure 2: *Sample ecosystem map for the Tor project.*

Appendix D: Ecosystem Mapping Worksheet

Open Source Ecosystem Mapping Worksheet

Version 1.0

Use this page to draw a map or directed graph of current and potential actors in your ecosystem. List service providers and group them by the type of service they offer. Identify potential collaborators, and mark the ones with competitive service offerings. Identify competing open source and proprietary substitutes for your open source project.

Place actors with large, current impact closer to the center of the map and future recruits further away. The open source project belongs at the center, and you yourself might be close in or further out, depending on its current effective scale of involvement and investment.

When done, note interesting relationships between various nodes on the map. Add customers in another color. This map is a picture of your world as it currently exists and how it might change in the near-term future. Be sure to save a snapshot of this map and see how it shifts over time.

Appendix E: Open Source Goal Setting Worksheet

Open Source Goal Setting Worksheet

Version 1.0

Overall Goals

In this column, describe your overall goals for investment in this product or technology.

Project Goals

In this column, describe the goals of the open source project (not just your organization's portion of the project).

Your Open Source Goals

On this page, circle or highlight up to three important open source goals from the list. Select up to 3 more secondary goals. Note them with a checkmark.

Development And Collaboration Goals

- Amplify or expand developer base
- Market and contextual insight
- Framework for partner collaboration
- Lead a standardization effort
- Disrupt an incumbent, hold off insurgents

External Marketing Goals

- Ease vendor lock-in fear
- Engage with users
- Transparency for customers and partners
- Establish a basis for product reputation
- Branding and credibility

Internal Goals

- Improve internal collaboration
- Improve developer hiring pool
- Improve morale and retention
- Innovation
- Improve open source capabilities