18F

U.S. Army ARL Path Analysis

Findings & Final Recommendations

May 3, 2019

Team

Andrew Dunkman, andrew.dunkman@gsa.gov
Kathryn Connolly, kathryn.connolly@gsa.gov
Eleni Gesch-Karamanlidis, eleni.gesch-karamanlidis@gsa.gov

Overview

We have spent eight weeks working with our partners at the U.S. Army Research Laboratory (ARL) investigating ARL's open source software needs and current process, with a goal of making recommendations to improve the current process workflow.

During the course of this Path Analysis, we generated the following findings and accompanying recommendations to prepare ARL for optimizations to their open source software process, as to be determined based on organizational priorities as discussed below.

Problem statement

ARL needs to be able to efficiently and transparently share open source software within legal requirements, but the current approach feels imbalanced.

Data

The findings in this report were generated from primary sources as well as secondary sources. We gathered the perceptions of 1 representative from each of the key stakeholder groups at ARL including researchers (civilian and contractor), OpSec, Legal, Tech Transfer, supervisors, Business Acumen and Invention Evaluation Committee. We also spoke with 1 Army Futures Command representative. For our secondary sources of data, we referred to publically-available Department of Defense, Department of the Army, AFC, and ARL statements or memos.

Terminology

For the purposes of this document, open source software includes all software and data for which the source is available for use, modification, and redistribution. The word software is used to represent a program which is used as a product or project, and the word code is used to represent a section of that software.

Open source software considerations

Common definitions

Our research into relevant policies and memos revealed multiple meanings for open source software. OMB M-16-21, the Federal Source Code Policy, looks to the Open Source Initiative and the Free Software Foundation to define a subset of licenses which published code must be released under for it to be considered "open source software" — whereas the Department of Defense, in it's 2009 memorandum Clarifying Guidance Regarding Open Source Software, defines open source as software for which the source code is available for use, modification, and redistribution — but does not specify license requirements. ARL's Legal team presents the definition from this Department memorandum in their presentation entitled "Introduction to Computer Software Protection and Release."

1.1. Recommendation

We recommend explicitly defining open source software as software for which the source code is available for use, modification, and redistribution in all communications and policies going forward. We believe determining a common definition will require ARL to do some further exploration into what are the current industry and government standard definitions, and which most aligns with what ARL is trying to accomplish. 18F partners with agencies like ARL to align diverse stakeholders around unified definitions of open source software, and identify the needs and tools most relevant and useful to their particular organizational context.

Without defining this term, communications and policies may be read to include or exclude software which is not subject to copyright, which includes works of government employees as per 17 U.S.C. § 105, as works that are not subject to copyright cannot be released under licenses which rely on copyright authority.

The lack of a common definition for open source software creates confusion when dealing with government software, and creates a risk of misunderstanding policies intended to cover all software to which the source code is available, regardless of the author.

Licensing and intellectual property

In open source software at ARL, determining authorship is difficult because involvement of contractors and university collaborators introduce a variety of present and potential rights because of the relevant laws and regulations. Thinking of these things upfront can make open source easier to do.

1.2. Recommendation

For projects that have not yet begun, we recommend taking a broader view of collaboration on open source software, starting from collaboration contracts such as Cooperative Research and Development Agreements. Revisiting these contracts will enable ARL to more easily embrace open source software in the future.

There is no clear person who should be thinking of software rights upfront when these agreements are signed, and as such software developed through these agreements without such a statement are legally complicated to publish, leading to a risk of increased exposure to lawsuits.

This legal complication delays collaboration which in turn slows down the pace of research at ARL, and involves multiple parties within ARL working together to sort out the intellectual property considerations.

Through our research, we've learned that ARL has both new software identified for potential open sourcing as well as existing software which it would like to publish as open source. The software has been developed collaboratively, with government and non-government contributors, and may contain code which has been released under open source licenses.

1.3. Recommendation

For existing software projects, we acknowledge that there is no simple answer which enables faster and less risky legal review. We recommend applying an open source software license as early in the development process as possible, as the legal review process is more simple with fewer contributors. For software which has not yet been written, this review process requires only selecting a legally appropriate license, simplifying future work and risk substantially.

Although the work of government employees is not subject to copyright as mentioned above, the joint work itself may include contributions from non-government employees and is subject to copyright. We recommend following the guidance provided by Code.mil's "How to Open Source" guide, including declaring the intent of the license for use in countries with different copyright laws. 18F itself uses a similar statement declaring unrestricted rights worldwide for countries where the work would be considered under copyright.

WIthout declaring such a license, open source projects increase the risk of exposure to lawsuits from external collaborators for the government.

When accepting contributions from outside collaborators, the default assumption in the open source community is that the inbound license is the outbound license — that the contributions given to a project are released for use under the existing license. ARL's current policy makes this assumption explicit by requiring all contributors to deliver to the project owners a signed Contributor License Agreement. ARL can more fully recognize the benefits it can achieve with open source software by more effectively leveraging existing software communities.

1.4. Recommendation

To accept contributions for projects which have been released as open source, we recommend removing the requirement for a signed Contributor License Agreement, which poses a high barrier to entry for new contributors.

To make the implicit assumptions about licenses explicit, we recommend instead relying on the terms of service of a code hosting platform or by using a Developer Certificate of Origin (DCO) process, which involves having a new

contributor review a statement of license terms and express agreement to it by adding their name and email address to a repository's contributing document.

To successfully encourage external community members, new collaborators must be able to easily suggest code for review and have approved code merged. Contributions to open source projects often begin with small typo-grade fixes and then after a contributor sees that a community is active, invests more in larger contributions of higher value. A contributor license agreement is a high barrier to entry for a community, and may significantly impact the number of people contributing and as such significantly decrease the benefits ARL realizes from that community.

If a project is hosted on GitHub, GitHub's terms of service already handle making the inbound license the outbound license, and no additional barriers are needed to be legally protected:

> "Whenever you make a contribution to a repository containing notice of a license, you license your contribution under the same terms, and you agree that you have the right to license your contribution under those terms. If you have a separate agreement to license your contributions under different terms, such as a contributor license agreement, that agreement will supersede."

If the project is not hosted on a site which makes this statement in their terms of service, or if additional assurance is needed in case the code is transferred to a different code hosting service, we recommend using a lightweight DCO process, as also recommended by Code.mil's "How to Open Source" guide and as is described there.

This process balances legal risk with contributor productivity, allowing assurance that the contributor has reviewed and agreed to the license terms explicitly without adding steps outside of the standard code suggestion and review process.

To fully recognize the benefits that ARL can receive from open source communities, it must lower the barrier for community members to enter to encourage volunteer participation.

Balancing risks

In our findings, we see that ARL acknowledges that there is a risk associated with sharing information, but it also realizes that open innovation is critical in yielding unpredictable and critically important results as proven through its Open Campus initiative. ARL recognizes the need to collaborate with others through academic publications and other means. It has recognized that collaboration is one way to access external resources that can help develop software faster, cheaper, and more efficiently. It also faces resource constraints, and needs to find ways to maximize effect towards its mission.

There is a spectrum of risk: on one end, maintaining strict security in sharing no knowledge, which ARL has recognized is not acceptable as it cannot maintain the Army's goal of near-peer overmatch. On the other end, sharing all knowledge, which would enable its near-peers a potential competitive advantage over its developing technology, which is also not acceptable.

To share with confidence means knowing what is safe to share, with whom, and how much. Right now, people at ARL don't feel that there is an easy answer — and to answer where on this spectrum of risk ARL should fall, the organization needs to assess its goals and values as it relates to open source software, so it can judge if it's sharing too much or too little and if its risk assessment processes are aiding or hindering its goals.

1.5. Recommendation

We recommend clearly defining goals for open source software and for contributions to open source software at ARL. If it fully recognizes the benefits of open source software participation, ARL must consider the impact of this participation when evaluating employee performance.

Defining how ARL will approach open source software will allow ARL to accurately make policy decisions to further that stated goal:

- Decide what should be considered a safe contribution to existing open source projects and does not require security review

- Determine if and how existing code should be published as open source, and how much time and effort is needed to do so
- Define what is considered successful, and judge in what ways open source projects are or should be expected to return the investment which ARL is putting in them

Without internal alignment on the benefits it wishes to see from open source software participation, ARL cannot accurately weigh the risks of sharing too much with the risks of losing near-peer overmatch. Without including the impact of open source software participation in employee performance evaluations, ARL is suggesting that open source software is not a valued contribution to mission objectives.

Our research has shown that ARL recognizes that software development moves at a pace that's quicker than what ARL's business processes or risk review can currently keep up with. We have learned that ARL recognizes that judgement calls and interpretations from those with determination authority are critical in safely open sourcing software. Finally, we've also seen that open source software benefits scientific and academic communities, and helps ARL build civilian-military relationships.

1.6. Recommendation

We recommend relying on an employee or contractor's operational security training when determining what is acceptable risk in public participation in discussion and code contribution with regards to open source software.

There is risk associated with engaging in public collaboration, but ARL has already recognized the benefits that such collaboration provides. It already requires operational security training for its employees and contractors to enable them to collaborate with the public in this way through mediums such as email. We have learned through our research that ARL has previously used a Scope of Understanding to define a project's boundaries to more clearly indicate the bounds of an open source project to further reduce the risk of exposure.

Additional review steps for public contributions to open source software, which may happen as often as multiple times a day, increases the barriers to

participation for ARL's employees and contractors, which in turn decreases ARL's benefits in participating in open source communities.

Tools and technology

We have observed that ARL's current open source software policy includes specific tools as a matter of policy, and that some of ARL's existing open source software communities do not use these tools. Those tools include, among others, version control systems and source control hosting systems (internal and public).

The context to which software is being developed determines the best tools that should be used for the job. To fully realize the benefits of open source software collaboration, projects must be able to engage with existing software communities, regardless of what tools those communities use.

1.7. Recommendation

We recommend removing tool and technology specific language from open source software guidelines and instructions, and instead allow the context in which software is being developed to determine which tools are most appropriate.

Recommendations to use particular tools and technologies can quickly become out of date, which require the policy to be updated — or worked around — whereas keeping tool-specific recommendations in a best practice guide can be updated more readily as changes are needed, without rethinking topics such as operational security.

The context to which software is being developed determines which tools are most appropriate, and a published policy that contains tool-specific language increases the risk of less appropriate tools being used.

Product management

We found that folks at ARL need guidance on open source software sharing and refer to the stated policy or find someone for ad-hoc advice. Without a central role with both authority and accountability for open source software, there's no clear route to resolve misunderstandings. Currently there are several folks providing leadership through official and unofficial contributions. We've also learned that ARL has filled support service positions on rotation in the past, and that services like the Business Acumen office were created once ARL Leadership learned of employee concerns with the quality of those support services.

1.8. Recommendation

If ARL considers open source software sharing a critical support service, we recommend having one designated administrator or owner empowered with authority and responsibility for its success and stability long-term. That designated owner would leverage product management methodologies and frameworks that can help create a cohesive vision for the open source process and a tactical execution path that will maximize value and mitigate risk. Product management leverages user centered design to realize value and iterative and incremental delivery that creates a foundation for ARL open source software that can respond dynamically to future needs.

A product manager would drive a correct product-market fit for the open source software process; ensuring that it speaks to users, stakeholders, technology, and mission objectives. Some of our findings speak to a future vision for open source at ARL, while others speak to potential ways to deliver on a vision. A product manager is responsible for creating a product team and developing a common vision that binds them together. We recommend first identifying the right product manager. Here are key questions to answer:

1. Who understands the vision of open source software at ARL?
2. Who has a solid understanding of the stakeholders, users, technology, and business needs and can translate between these groups?
3. Who is empowered to prioritize and define the work for the product team in the long-term?

4. Who can set and measure progress against clear performance indicators, and communicate progress to stakeholders & management?

Answering these questions would help ARL identify whether open source software is a directorate-owned or enterprise-wide endeavor. By starting with identifying a product manager, ARL would establish a clear owner tasked with focusing ARL's future open source software efforts and maximizing the value from those efforts.

18F can help ARL answer these questions. 18F's product management team can work with ARL's current open source software process administrator to establish familiarity with product management concepts and frameworks which can be put into practice immediately. This training can be completed within the current project's budget in the immediate future.

Building off of this training, 18F can also partner with ARL to outline the skillset and talent requirements for a product manager based on its current and anticipated open source software usage, needs, and priorities. Our product management team works with partners like ARL to identify potential candidates and their related offices to take on the manager role, as well as clarify the ecosystem of stakeholders critical to supporting this support service.

We believe this would prepare ARL with a solid foundation for follow on discussions within the organization of roles and responsibilities. This would also mitigate current confusion among people who are assigned a role in the process but don't always know what it is.

We also believe the upfront costs associated with creating a product manager role are far less than the costs of current confusion and safety assurance on ARL's workforce productivity. Iterative work environments are productive because the workforce is supported by clear sources of stability. ARL's ability to open its doors to external collaborators safely and efficiently requires that all key players are on the same page with a unified vision and implementing practices. There is currently a high risk to ARL's ability to share open source software and scale to future requirements. Doing so safely and efficiently means diverse experts aligning their independent work around a common goal. If someone's go-to advisor isn't available or no longer were to be at ARL, there is a strong likelihood of them getting stuck or avoiding sharing. Thus, these implicit leaders are currently more important to the process' success than the overall team.

Independent evaluation

Organizational structure & priorities

To meet the needs of the Army of the future, people at ARL feel they need to be able to collaborate with external groups, even if that means overcoming a “we don’t do it that way” mentality — yet there seems to be a disconnect within ARL as well as between ARL and the Army Futures Command (AFC). It’s not clear to folks in both ends of the vertical who else exists, their responsibilities and priorities, and how they do their job. It’s also not clear to folks how information flows across these groups, though we learned there are current information sharing practices that are considered user-friendly. It’s also not clear when authority is more appropriate centralized or decentralized. We learned that some people feel Code.mil’s lack of traction with their proposed Department of Defense open source software policy and ARL’s ability to establish ARL’s open source software policy are two examples of unsuccessfully and successfully navigating this tension.

1.9. Recommendation

We see a strong need for ARL and AFC to have greater organizational awareness of the people, mission objectives, needs, and tools that make those organizations run smoothly. The Army has a bold vision for how it will fight in the future. In 2028, the Army will be part of Multi-Domain Operations, to fight by sea, space, cyber, land, air at the same time. To determine how best to prepare for this future battlefield, the Army created AFC in 2018. AFC’s General Murray said at the 2019 AUSA Global Force Symposium, “you can’t know where you’re going until you know where you’re at.” An organization knowing where it’s at includes knowing what it has.

We recommend AFC and ARL take stock of who they have, what those groups are working towards and why, and what tools they need to accomplish their mission. There are immediate and long-term gains for both ARL and AFC and we think ARL could pursue this recommendation independently or as part of larger Command-wide effort.

ARL taking stock of the organization would enable leadership and enterprise-wide groups, such as Business Acumen, to identify existing strengths, opportunities for leveraging and multiplying those strengths, as well as prioritizing improvement efforts.

AFC would greatly benefit from determining downtrace needs, capabilities, and tools across the nascent Command as it ramps up it's newly-organized workforce in several ways:

1. Determine how to offer maximum flexibility to downtrace within legal requirements, and where centralized authority is appropriate
2. Identify strategic challenges and opportunities connected to the Army's Multi-Domain Operations Doctrine to design into the Command's emerging organizational chart
3. Identify needs for a unified data environment and whether a cloud environment might improve leadership's access to data for modernization effort prioritization decisions
4. Determine whether open source software is an effective leverage point to accelerate the development of new civilian-military relationships

Knowledge transfer

Through our research we found that ARL does not have a place to store, search, and access knowledge which meets the needs of its employees and contractors. People resort to informal information sharing like searching email and searching out subject matter experts, which requires more time, effort, and has the potential to exclude part of the workforce.

We also learned that it's a priority to ARL that the knowledge it generates influences leadership decisions, and that success stories are one way of accomplishing that goal. Folks at ARL feel they don't have the communication tools they need to do their job well, and have successfully found ways to overcome this challenge by using emails, aggregating and tracking data in spreadsheets, and using Sharepoint sites, even if they require more manual effort.

1.10. Recommendation

We recommend finding the lessons learned in open source software at ARL and sharing them broadly outside of the organization.

Effectively sharing lessons learned and ARL's open source software success stories can break down silos within ARL and cut through vertical layers within the Army to showcase ARL's efforts with AFC and beyond. Focusing on these successes can position it as a leader in open software collaboration, amplifying the benefits it can realize from open source software collaboration as well as influencing ongoing modernization efforts within the Army.

Without sharing ARL's success stories related to open source software, those lessons learned are not widely known, and others attempting to open source code are likely to make similar mistakes, decreasing the returns on investment that ARL is making in open source software. ARL's success stories are a positive influence within AFC and beyond, and not sharing them is a missed opportunity to influence leadership decisions.

Define process

We interviewed representatives from key groups within the open source software process as well as representatives who have completed this process to generate a Process Map (Perceived), demonstrating the steps encountered by a researcher attempting to collaborate with open source code.

Additionally, based only on ARL's written Open Source Guidelines and Instructions, we generated a second Process Map (Intended) which demonstrates the steps to open source software as written.

The policy is intended to give researchers agency to open source software, but our research shows that the process prevents them because of the effort required. We have learned that some parts of ARL's current process work smoothly, and some parts are confusing or worked around.

1.11. Recommendation

We recommend taking a holistic approach to process optimization, including examining how the steps in the Form 1 process associate with the open source software release steps, and writing guidance on how to appropriately complete Form 1 for software releases.

Our research shows that some of the software release steps are duplicated effort with review steps from Form 1, and our research suggests these steps exist because there is uncertainty that what was reviewed in Form 1 is the same as what is now being reviewed for software release.

Additionally, our research suggests that the people involved in ARL's software release process do not believe there are too many people involved — but rather they recognize the benefits of each review step however the overall time and energy required to complete the process is unreasonable.

WIthout directions in how to complete ARL's Form 1 process for software releases, people involved in the additional software-specific steps do not have confidence that the software itself was sufficiently reviewed, which requires double-checking of the previous review steps, leading to delays and a feeling of backward momentum.

People involved in reviewing and approving software have been assigned a role but they don't necessarily know what it is or how to accomplish it, and those wanting to release code enter into a process where they can't tell where they're at and when they're done. They understand that ARL's open source software process is still being developed.

1.12. Recommendation

We recommend clarifying the steps involved in publishing code as open source, and then exploring training and educational opportunities to walk people through what to expect and what is expected of them.

With greater understanding of the process as a whole, each review step along the way can be optimized to remove duplicated efforts, as the system as a whole has greater confidence in itself.

Without clarity in the responsibilities involved by each party in the software release process, there is duplicated effort causing delays and discouragement, which prevents ARL from effectively realizing the benefits of open source collaboration.

Roadmap strategy

Strategic roadmap of opportunities

18F was engaged in this Path Analysis to address how to balance ease of use with legal requirements in ARL's open source software policy & process. This appears to be the most easily visible, but not the only, key question for ARL to answer. We identified a series of critical upstream questions impacting the current workflow that if left unanswered, will continue to drive perceptions that the current workflow is problematic. ARL will not be able to assess how big of a pain point open source software really is, whether it's worth solving, to which extent, and in which way without first addressing these questions.

1. Even with a streamlined process map, what is still unclear is whether there is broader organizational alignment on who are the right people as well as what are the responsibilities, steps, and tools needed for the overall workflow to run effectively.
2. Even with alignment on the overall workflow, what is still unclear is where ARL stands on sharing open source software in various contexts shaped by risk, authorship, and software age factors. Folks think there is a risk associated with sharing information. To share with confidence means knowing what is safe to share, with whom, and how much. Right now, people at ARL don't feel there is an easy answer.
3. These previous questions are based on ARL's current needs, they do not consider how ARL's open source software needs will evolve into the future.
4. Even after considering ARL's future open source software needs, what is still unclear is how important it is to organizational priorities, and how much software

sharing will meet organizational goals. Open source software doesn't appear to be a top organizational priority.

1.13. Recommendation

In our independent evaluation, we've laid out several recommendations for developing a streamlined open source software process. 18F was engaged because of perceptions in the organization that the current process workflow was problematic. We see a risk that focusing only on process optimizations would not resolve concerns with the process itself, because process design itself is not the only source of those concerns. We believe the process is perceived as problematic due to a lack of alignment across the organization on several fundamental questions, which we strongly recommend ARL align on before pursuing any process improvements.

Specifically, we suggest internal discussions to align on these topics:

1. How does open source software achieve ARL's mission objectives?
2. How much and what type of sharing is needed for ARL to meet its mission objectives?
3. What, and for whom, are the specific benefits of open source software to ARL?
4. How much time is required, and by whom, to develop and maintain open source participation?
5. How much time is required, and by whom, to develop and maintain collaborative relationships or feedback loops with the public?
6. How much time is required, and by whom, to maintain the systems and code generated from open source software?
7. How and at what time does ARL retire existing open source projects, and based on what metrics?

Because of the distribution of expertise within ARL, we recommend those discussions involve leadership as well as all key groups involved in the process, as noted in the Process Map (Perceived) generated through this engagement.

The outcome of those discussions will provide ARL clarity on whether open source software is worthwhile to the organization, and if so, to what extent and in which contexts. We note that ARL can align on these discussions and create greater certainty within the organization even as uncertainty remains regarding DoD's and AFC's policy on open source software.

It may be that open source software is not deemed to be a top priority for ARL at this time. In that case, ARL would be better served focusing their process improvements elsewhere, and engaging the Business Acumen office to determine which business processes align most with ARL's current focus areas. In regards to the open source software process, we recommend several quick fixes to the current process that can be achieved in the short-term with minimal resources:

1. Revise the Written Policy to separate out policy from procedures. The policy section should clearly state objectives and policy. The procedures section should specifically lay out step by step the process in its entirety; what must be done and by whom, starting with steps to be taken during the Form 1 process.
2. Revise the Intended Process Map as needed to accurately reflect the revised policy & procedures.
3. Post the Process Map throughout ARL and ALC in key stakeholders' work areas.
4. Engage Tech Transfer office to develop a success story about this engagement to share within ARL and AFC.
5. Post relevant documents on ARL Inside and ARL's GitHub repository that includes
    a. Policy & Procedures document, to lay out goals and steps
    b. Process Map (Intended), to set expectations on timeline and steps needed to complete the entire process
6. Communicate to all branch chiefs what documentation exists and where to find it.

If open source software is deemed to be a top priority, ARL will have resolved much of the sources of confusion on the current process and be in a much more strategic position to prioritize the improvement opportunities around process roles, responsibilities, steps, and tools, which we have laid out in our independent evaluation.