18F

U.S. Army ARL Path Analysis

Findings & Final Recommendations

May 8, 2019

Team

Andrew Dunkman, andrew.dunkman@gsa.gov

Kathryn Connolly, kathryn.connolly@gsa.gov

Eleni Gesch-Karamanlidis, eleni.gesch-karamanlidis@gsa.gov

Overview

For the past eight weeks, we've partnered with the U.S. Army Research Laboratory (ARL) investigating ARL's open source software (OSS) needs and current processes. The findings and recommendations developed during the course of this Path Analysis, are aimed at preparing ARL to optimize their open source software processes, based on organizational priorities. Our findings and recommendations are divided into four focus areas:

1. Open source software considerations
2. Product management
3. Independent evaluation
4. Strategic OSS roadmap

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 both primary sources and secondary sources. We gathered the perceptions representatives from each of the key stakeholder groups at ARL including researchers (civilian and contractor), OpSec, Legal, Tech Transfer, research supervisors, Business Acumen and Invention Evaluation Committee (IEC). We also spoke with a representative from Army Futures Command (AFC). Secondary sources of data included 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.”

Recommendation

We recommend aligning on a common definition for open source software. Specifically, we recommend 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.

This definition includes software which is not subject to copyright, which includes works of government employees as per 17 U.S.C. § 105. We recommend using this definition because whether or not the software is subject to copyright is not relevant to ARL determining the appropriate process to release it, with the exception of attaching the appropriate software license to the code.

Because there is no common definition of this term, we recommend including the intended definition in all policies, practices, agreements, and communications going forward.

The continued lack of a common definition for open source software would create further confusion when dealing with government software, as well as increase the likelihood that approvers and reviewers will misinterpret policies intended to cover all software to which the source code is available, regardless of the author.

Licensing and intellectual property

Our research suggests that 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 issues, due to various laws and regulations. The current requirement for a signed Contributor License Agreement poses a high barrier to entry for new contributors.

Recommendation

Thinking of these challenges upfront can make open sourcing software easier. For projects that have not yet begun, we recommend making changes to the collaboration agreements that will define those projects. Including open source software rights in CRADAs will enable projects to be open sourced at a later date more easily.

If ARL continues to structure CRADAs as it currently does, these collaboration agreements will continue to be signed without upfront thought paid to software rights, which means ARL will continue to face legal complexities when trying to publish developed software. These legal complexities delay collaboration which in turn slows down the pace of research at ARL, because of the multiple ARL stakeholders who need to be involved to sort out intellectual property considerations. And yet, ARL's risk of potential future lawsuits remains. We anticipate acting on this recommendation would require Tech Transfer to work with ARL's Legal team to make changes to their existing CRADA templates, and to negotiate these statements with collaborators per contract as needed. However, those costs are significantly outweighed by the benefits of significantly reducing the risk of future lawsuits related to ARL software rights.

Recommendation

ARL can more fully recognize the benefits it can achieve with open source software by more effectively leveraging common open source software practices. To accept contributions for projects which have been released as open source, we recommend using a method for making implicit assumptions about licenses explicit without requiring specific tools or breaking outside of the tools used for software collaboration.

Two such ways to do so are to either rely on the terms of service of a code hosting platform or to use a sign-off process within the repository by using a Developer Certificate of Origin (DCO).

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."

Alternatively, a sign-off process within the repository with a DCO is more similar to a Contributor License Agreement — it requires contributors to review a license statement and add their name to a contributors list within the repository. This recommendation is in line with Code.mil's guidance, which enables easy collaboration with other Department of Defense (DoD) organizations following the same guidance.

For projects hosted on code hosting platforms with contributing license statements such as GitHub, we recommend using no DCO or CLA, as the terms of service already address these issues. For projects not hosted on these platforms, we recommend using the DCO process to balance 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 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.

Balancing risks

In our findings, we see that ARL acknowledges that there is a risk associated with sharing information. 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.

We learned that stakeholders view managing risk as a balancing act. To share with confidence means knowing what is safe to share, with whom, and how much. If ARL doesn't share any knowledge with external collaborators, they wouldn't be able to keep up with the pace of technological advance on their own and would not be able to reach the Army's goal of outpacing it's enemies, or "near-peer overmatch." On the other hand, if ARL's external collaboration is not limited in some way, the wrong information could get into the wrong hands and threaten the Army's competitive advantage over those near-peers.

Right now, people at ARL don't feel that there is an easy answer. We have learned that for some open source projects, ARL relies on a Scope of Understanding to define a project's boundaries to more clearly indicate what kind of collaboration is considered low risk; in ARL's current open source software policy, it indicates that Operation Security review is required for all contributions; and in other forms of outside communication such as email, ARL relies on training to mitigate risk.

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 adapt to. 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 learned that perceived benefits of open source software sharing include collaborating with scientific and academic communities, and building civilian-military relationships.

Recommendation

We recommend relying on training in order to make judgement calls when determining acceptable risk in public participation of open source software discussion and code contribution instead of requiring Operational Security review for all contributions.

To correctly make difficult judgement calls, understanding of the code and context around a project is required. ARL should first identify who are those with the expertise and authority to assess risk for a given project. Then, the fundamental topics of knowledge to ground their judgement calls needs to be identified. Anyone who's job entails making these risk assessments should receive the same training on this foundational knowledge. We recommend ARL identify within the organization how to create and approve this training curriculum. Once a training is in place, a process would have to be established for ensuring that moving forward, appropriate individuals are identified and trained. In an environment where the risk review process may be as frequent as multiple times a day, having a standard starting point across individuals with different interpretations and judgement is critical for ARL to safely determine acceptable risk.

If operational security review is required for public contributions, the barrier to participation in open source software for ARL staff risks defeating the purpose of open collaboration by effectively preventing participation.

Tools and technology

We have observed that ARL's current open source software policy references specific tools while also suggesting individuals are free to choose what tools they use. Also, we learned 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), such as GitHub.

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.

Recommendation

When people read ARL's open source policies, they may perceive a preference for particular tools which may not be the most useful choice for their project. We recommend separating ARL's existing policy into two documents, one focused on policies and the other on procedures.

Within the policy document, we recommend addressing the source of each policy section, especially for sections which realize federal or Army laws and regulations. This will enable greater understanding of where possible changes can be made more easily going forward.

Within the procedures document, we recommend identifying that the tools chosen are generally the most appropriate to comply with the stated policy, but that others may be more contextually appropriate. This is similar to 18F's open source practices, which implement 18F's open source policy (which in turn implements GSA's open source policy).

Separating out the policies and procedures in this way enables flexibility in choice of tools within the bounds of compliance with policy, as well as enables updating the procedures documentation as better tools and procedures are available without rethinking topics such as operational security.

If procedures are not separated from policy, there is a risk that software that would benefit from open source collaboration but does not use the tools referenced is not open sourced, or that contextually inappropriate tools are chosen, reducing the benefits that ARL would see. There is also a risk that the current policy becomes outdated but there is no clear path to perform tool-specific updates without a full review of the policy, which may be difficult for the organization to perform.

Product management

We found that stakeholders at ARL require guidance on open source software sharing and refer to the stated policies or seek out 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 people 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.

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 ensure that the process 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.

There is a high risk to ARL's ability to share open source software and scale to future requirements, if ARL continues with a current temporary administrator role. Releasing code safely and efficiently requires diverse experts to align 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 will continue to be more important to the process' success than the overall team if nothing changes.

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.

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. Our research uncovered a disconnect across ARL and AFC. We discovered a lack of clarity around roles, responsibilities, and clearer directives on how people work. Information flow is also lacking between groups, and we heard that existing information sharing practices are not considered user-friendly.

It's also not clear when authority is more appropriate centralized or decentralized. Additionally, we learned that some people feel Code.mil's lack of traction with a proposed DoD open source software policy as compared to ARL's ability to establish an open source software policy are examples of opposing ends of this tension.

Recommendation

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 assess what makes their Command, and downtrace units, run smoothly.

Specifically, we recommend that AFC conduct broad assessments of the entire Command. At the ARL level, we believe this assessment would be led by leadership and enterprise-wide groups such as the Business Acumen office, to assess the organization across and within Directorates. Broadly speaking, AFC's assessment should include these key areas:

1. Current business needs based on Army mission objectives
2. Existing strengths and successes, and key driving factors
3. Existing tools & solutions that support business needs

The scope of this recommended assessment is broad by design, to generate a comprehensive internal data set that leadership can repeatedly leverage in a broad array of organizational decisions. AFC could take on this assessment as a stand alone project. Or, this assessment could be conducted during the course of more specific or focused projects.

For example, during the course of our research, we identified the following organizational challenges that may be potential or emerging projects within AFC. Such specific projects would serve well as vehicles through which to conduct our recommended organizational assessments:

1. Designing and implementing a unified data environment at AFC for improved leadership access to data for decision-making
2. Designing the Command's emerging organizational chart in line with the Army's Multi-Domain Operations Doctrine
3. Determining how to offer maximum flexibility to AFC's downtrace within legal requirements, and where centralized authority is appropriate
4. Defining AFC's approach to open source software and its usefulness towards accelerating the development of new civilian-military relationships

As AFC matures as a new Army Command at a rapid pace, we see a risk to leadership's ability to make effective decisions without an accurate picture of what's downtrace. If AFC makes critical design decisions regarding Command structure, policy, and principles without knowing what organizational assets it can leverage, leadership are acting without the best information at hand. If the Army forces are going to "militarily compete, penetrate, dis-integrate, and exploit our adversaries in the future" through Multi-Domain Operations, they're going to need more than rapidly evolving technology. They're going to need to leverage their assets, the people who make up the Army Force. That requires having an accurate understanding of the current state.

Knowledge transfer

We have learned that it's a priority to ARL that the knowledge it generates influences leadership decisions, and that ARL staff feel they lack communication tools necessary to do their job well. Through our research, we have also been sent a document titled "Guidance on Open Source Participation," which was generated by a researcher at ARL, and appears to indicate a duplication of effort with this initiative.

Recommendation

We recommend developing a specific pipeline for discovering and sharing lessons learned in research and open source software collaboration across and outside of ARL. We have learned that one such way to do so is by sharing success stories through ARL's Tech Transfer office.

It is not clear to us that ARL is used to sharing information in this way. We recommend developing a success story about this engagement as an opportunity to test out this pipeline as a mechanism to 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.

If ARL does not have a specific pipeline to discover and share its lessons learned, there is a risk that it is not known for the research it has developed and that it does not have the chance to influence leadership decisions. There is also a continued risk in duplicated efforts across the different silos within ARL.

Defined 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 perceived OSS process workflow, 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 an intended OSS process workflow 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 stakeholders view ARL's current process as not fully developed and confusing. To resolve issues in the process, 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.

For example, assigned reviewers and approvers aren't always aware of their role in the process while researchers don't know where they're at in the process. Thus, stakeholders have resorted to work arounds like sending emails and attaching printed code as an appendix to a research paper to release code.

Our research shows that some of the software release steps are duplicative with Form 1 review steps, causing confusion during the review process, depending on what was initially submitted during Form 1.

Recommendation

As recommended above in *Tools and technology*, we see benefits here in separating ARL's open source software policies into Policies & Procedures as well. In the Procedures, we recommend specifically laying out the step by step process in its entirety: what must be done and by whom, starting with steps to be taken during the Form 1 process.

This Procedures document would then serve to accurately depict the steps of the process in reality, and would serve as a reference for indicating the level of effort required for, and the role of, researchers, reviewers, and approvers.

Further delay in making this process more clear would amplify the effects that we have already seen in our research — that the uncertainty and confusion in the process is a deterrent to open source collaboration.

Strategic OSS roadmap

We organized our project efforts around the question of how to balance ease of use with legal requirements in the process' implementation. Our research suggests this to be the most easily visible, but not the only, key question for ARL to answer. We identified stakeholders' lack of clarity on a series of critical upstream questions negatively impacting their experiences navigating the current workflow as well. Our findings led us to wonder just how much of a priority OSS is to ARL's current priorities, and given the current low level of interest we gauged, question the usefulness of investing resources to improve the current process. Our findings suggest that lack of clarity on more fundamental issues around ARL's approach to OSS is having a great impact on the current process than the specific manner in which it is being implemented.

1. Even with a streamlined process map, ARL lacks broader organizational alignment on relevant key stakeholders, responsibilities, processes, and tools needed for the overall workflow to run effectively.
2. Even with this alignment, moving upstream, ARL needs to define a policy on tands on sharing open source software in various contexts as shaped by risk, authorship, and software age factors. Stakeholders perceive risks associated with sharing information. For them, sharing 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 questions are based on ARL's current needs, they do not consider the more upstream consideration of how ARL's open source software needs will evolve into the future.
4. At present, open source software doesn't appear to be a top organization priority. Further upstream for ARL to determine is how important and to what extent, OSS will meet organization goals.

While these questions remain unanswered, it is unclear how ARL would best prioritize the recommendations we've outlined in our Independent Evaluation.

Recommendation

Before pursuing the process improvements outlined in our Independent Evaluation, we strongly recommend ARL first align on the following questions:

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 when 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. The intended process workflow visual created through this engagement notes the key stakeholder groups identified through our research and would be a good starting point for further validation of the identified groups by more members in the organization than were sampled in our research. We should 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. The outcome of these discussions would provide ARL with clarity on whether open source software is worthwhile to the organization, and if so, to what extent.

If ARL determines that OSS is not a top priority at this time, we have two specific recommendations. First, ARL should focus their process improvements elsewhere, and engage enterprise-wide groups, including the Business Acumen office, to prioritize process improvements efforts in alignment with ARL's current focus areas. Second, ARL can improve the current process in the short-term with minimal resources with these quick fixes:

1. Revise the Policy documentation into Policy & Procedures written documentation that separates policy from procedures, as detailed in *Tools and technology* and *Defined process*
2. Revise the intended OSS process workflow as needed to accurately reflect the Policy & Procedures documentation
3. Display the updated intended OSS process workflow in all key stakeholders' workspaces
4. Engage Tech Transfer office to develop a success story about this engagement to share within ARL and AFC, as detailed in *Knowledge transfer*
5. Post relevant documentation on ARL Inside and ARL's GitHub repository that includes
    a. Policy & Procedures
    b. Intended OSS process workflow
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, we have two specific recommendations. First, ARL should implement the above quick fixes to ensure that current process is communicated clearly across the organization. Second, internal alignment on ARL's approach to OSS will very likely resolve much of the confusion surrounding the current process, placing ARL 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.

We see a big risk in delaying on this recommendation. The critical upstream questions we identified, if left unanswered, will continue to drive perceptions that the current workflow is problematic. Therefore any process improvements tackled before answering those questions are likely to be unsuccessful as they will not be tackling the issue at its root. ARL would be investing resources in a business need that is potentially lower priority than more pressing impactful areas of its mission. ARL would be without critical information to weigh the risks and make an effective decision as to how big of a pain point open source software really is, whether it's worth solving, to which extent, and in which way.