

December 13, 2021

Office of Advanced Scientific  
Computing Research (ASCR),  
Office of Science, Department of Energy.

RE: "Stewardship of Software": DOE\_FRDOC\_0001-4278

To Whom It May Concern:

The Incorporated Research Institutions for Seismology (IRIS) hereby submits these comments in response to the Office of Science (SC) in the Department of Energy's (DOE) Request for Information regarding the stewardship of the software ecosystem for scientific and high-performance computing.

IRIS is a consortium of over 100 US universities dedicated to the operation of science facilities for the acquisition, management, and distribution of seismological data. IRIS programs contribute to scholarly research, education, earthquake hazard mitigation, and verification of the Comprehensive Nuclear-Test-Ban Treaty. IRIS is largely funded through the Seismological Facilities for the Advancement of Geoscience (SAGE) Award of the National Science Foundation under Cooperative Support Agreement EAR-1851048.

## INTRODUCTION

The Data Services directorate of IRIS (IRIS-DS) has been the primary steward of software and services in support of IRIS's mission to facilitate the flow and caretaking of high quality digital seismic data to researchers and computing facilities around the globe. In addition to creating tools and services in-house, IRIS-DS also collaborates with an international community to create standards for both data representation as well as software behaviors. IRIS-DS also provides discovery and access to contributed open-source software maintained by community members, thereby permitting collective sustainment of these technologies.<sup>1</sup>

---

<sup>1</sup> <https://ds.iris.edu>

For the past decade, IRIS Data Services has adopted and developed a much more service-oriented model in the way it delivers data to clients and provides a wide array of data discovery and value-added high-level data products for interactive and programmatic access. IRIS Data Services has more than 30 web service endpoints that serve the needs of seismic data delivery, data transformation, data quality assurance, and supporting metadata of instrumentation and seismic events.<sup>2</sup> The programmatic performance of these services has been proven to fit the needs of high-throughput computing (HTC) and has permitted high volumes of data to populate experiments in high-performance computing (HPC) centers around the globe.<sup>3</sup>

In all of this, IRIS has grown increasingly dependent on solid and agile software repositories and package managers on the Internet to maintain both rapid software development practices as well as take advantage of a rich and evolving open source environment to solve current-day problems with data interoperability and scaling of data processing. IRIS being a contributor to the open-source community also highlights the investment we have in the structures of sustainable software ecosystems and to see these ecosystems grow with community input.<sup>4</sup>

## 1. SOFTWARE DEPENDENCIES AND REQUIREMENTS

IRIS Data Services leans very heavily in enterprise software development to support its service infrastructure. Applications are also developed for public use. Some are distributed for download but many reside as web applications. The languages and their dependencies typically in use can be summarized as the following:

- a. Java - Spring, Spring Boot, Maven, Tomcat, DAO, JPA
- b. Python - Django, Numpy, Pandas, ObsPy, Anaconda, Pip
- c. C - CentOS packages
- d. R - tidyverse, CRAN
- e. JavaScript - Vue.js, npm, D3
- f. Perl - DBI, LWP, CPAN

This list is representative of our language tools, even as the frameworks and dependencies listed here are nowhere near exhaustive. In many of these, a critical package manager exists to supply ubiquitous, versioned libraries that serve to solve a particular need. The wide availability and selection of these libraries invariably helps the developer to achieve their project goals faster and

---

<sup>2</sup> <https://service.iris.edu>

<sup>3</sup> MacCarthy, J., O. Marcillo, and C. Trabant (2019), Putting the cloud to work for seismology, *Eos*, 100, <https://doi.org/10.1029/2019EO119741>. Published on 05 April 2019.

<sup>4</sup> <http://ds.iris.edu/ds/nodes/dmc/software/>

with greater reliability. The key factor is that the developer knows that the package exists and can easily access it in their build routines.

In the future, we envision that we will be ever more dependent on reliable access to core packages to construct new technologies. Indeed, as we begin our designs to move to a containerized, cloud-ready architecture, that need will have never been greater. Our software will need to be able to carry out modularized continuous integration (CI) and continuous deployment (CD) processes with automated reliability. Below, we list frameworks that will serve the needs of being able to run our data facility in a cloud-ready fashion:

- a. Containers - Docker, OCI
- b. Container Management - Kubernetes
- c. Async Messaging - Kafka, RabbitMQ
- d. Serverless Functions - Kubeless, Spring Cloud Function, Zappa
- e. Observability - ELK Stack, Prometheus, Grafana

Having a DOE led pilot effort to help government-funded institutions find good software libraries and solutions (DOE pilot) for transforming to a cloud-ready facility would be of immense value. We view the cloud environment as a platform that will enhance both the scalability of our operations as well as the possible locality of compute to the data, by virtue of availability zone multitenancy.

## 2. PRACTICES RELATING TO THE SECURITY AND INTEGRITY OF SOFTWARE AND DATA

The security and integrity of software and data must be assured through careful practice at a number of levels of software development, testing, and deployment. While certainly the best first line of defense is careful programming practices on the part of the developer, what will be of critical concern to the DOE pilot is what happens after that software is written and transferred to source code control.

Clearly, what engenders the most confidence in the shared software community is self-verification of the code through comprehensive coverage of unit tests, mock tests, and regression tests. It is common to see open source projects on GitHub and elsewhere with tags that indicate their status as far as code coverage and test completion. This is indeed helpful, and good test verification during code check-in is critical, but the tests are only as good as the developer(s) volunteer to inject into the code. A recommendation to the DOE pilot would be to find a process to independently verify the efficacy of the test code shipped with the software.

A good example of software integrity through testing is the R language community's CRAN package manager.<sup>5</sup> Being sticklers for defect-free code, the CRAN repository collects submitted packages and their updates and runs test builds on their own systems, a variety of different platforms, to make sure they build and pass their tests. Any variations from a complete pass of these tests causes the package to be rejected until the fix is addressed. The value of such a repository is that all developers have a definitive benchmark of quality and interoperability that their code and documentation must meet before it will be accepted into general circulation. The DOE should consider whether it is feasible to establish a similar rigorous gatekeeping function.

Another need that comes to mind as relates to the integrity of any shared open source repository are measures that prevent supply-chain injection of malicious code and/or data into a package as it is assembled for release. As we witnessed in the SolarWinds hack<sup>6</sup> (and the less well-known ua-parser-js compromise via npm<sup>7</sup>), such an act can have devastating downstream effects on anyone using the compromised software or library. Not only can this destroy critical work or encrypt workstations, but it can serve as a trojan allowing undesirable access to internal systems. This point leads us to a final item of importance that should be carefully looked at.

The SolarWinds hack has been characterized by some as an attack on the identity management system.<sup>8</sup> Hackers were able to gain access to a commercial code base by use of a stolen token and an easily bypassed the multi-factor authentication (MFA) stage. The code was then compromised with malicious code that permitted the hackers to gain access to systems of any customer that used it. Our recommendation to the DOE pilot is to ensure that special attention is paid to implementing solid identity management and authorization techniques where the integrity of distributed code is concerned.

As a point of information in this comment, IRIS implements identity management for software access only through the distribution pathways it supports. These would be Django<sup>9</sup>, GitHub<sup>10</sup>, and RedMine (SeisCode)<sup>11</sup>. Such identity management is mainly used for maintainers as software downloads is mostly anonymous. IRIS distributes software mainly through these three

---

<sup>5</sup> <https://cran.r-project.org/>

<sup>6</sup> <https://www.businessinsider.com/solarwinds-hack-explained-government-agencies-cyber-security-2020-12>

<sup>7</sup> <https://www.truesec.com/hub/blog/uaparser-js-npm-package-supply-chain-attack-impact-and-response>

<sup>8</sup> <https://venturebeat.com/2021/11/18/top-lesson-from-solarwinds-attack-rethink-identity-security/>

<sup>9</sup> <http://ds.iris.edu/ds/nodes/dmc/software/>

<sup>10</sup> <https://github.com/iris-edu>

<sup>11</sup> <https://seiscode.iris.washington.edu/>

avenues, though it has published to the external repositories CRAN<sup>12</sup>, Maven<sup>13</sup>, and PyPi<sup>14</sup> as well.

### 3. INFRASTRUCTURE FOR SOFTWARE DEVELOPMENT AND DEPLOYMENT

IRIS has developed a fairly robust continuous integration (CI) system for many of its distributed and internal libraries and applications. Jenkins has been the predominant tool used for automated build and test operations endemic to a good CI process, though some use of Travis CI has been employed for projects residing on GitHub. Typical tests conducted at build time are unit tests and mock tests, followed by a snapshot distribution being generated for further evaluation. Good code versioning, branching, and merge practices for source code are integral to this process and is essential to an ordered rollout of feature changes.

IRIS has thus far not permitted hands-free automated deployment of successfully tested software but keeps a human in the loop to carry out application-specific regression tests and performance tests in a beta deployment environment that largely simulates the production zone.

Beta and Production deployments largely differ by the location of deployment of a software package along with a configuration that is specific to that environment. These deployments can be, and sometimes are, scripted to accommodate scalability and consistency in the release operations. Deployment scripting also helps ensure that multiple staff members have the capacity to release a patch or fix into production absent the primary developer.

Our recommendations to the DOE pilot would be to address the three areas listed above (automated tests, regression and beta tests, and automated deployment) for a robust archive of packages that have the stamp of approval by not only the developer but by DOE target systems as well.

Most especially for the latter case, where deployment test and verification processes are carried out, a validation model similar to that demonstrated by CRAN, where various platforms are instantiated and builds are deployed to them for standardized testing, would go a long way toward having verifiable processes that ensure software quality and broad platform availability.

Extending the level of validation in service of the high-performance computing community would be to run platform tests against moderate and large data problems to establish benchmarks and assess stability under load. The degree to which large-scale testing impacts production infrastructure must be assessed

---

<sup>12</sup> <https://cran.r-project.org/>

<sup>13</sup> <https://central.sonatype.org/>

<sup>14</sup> <https://pypi.org/>

and proportionately regulated to scheduled testing times. Using standardized workloads, code under benchmark evaluation could be inserted into the production queue and then off ramped to a standby location to evaluate metrics, outputs, and log files for final assessment.

While IRIS does not have immediate needs for HPC-oriented software and libraries, we do very much need to support the HPC community in their efforts to process vast amounts of seismic data, most especially through high-throughput modes of data access. For this reason, ready access to HPC codes developed by the community could prove very instructive for IRIS's future aims.

#### 4. DEVELOPING AND MAINTAINING COMMUNITY SOFTWARE

When it comes to maintaining open source software in service to the community, IRIS has been undergoing a notable change in the past five years in transparency and access to in-development software stacks through public cloud source code control repositories like GitHub. The benefit of GitHub is that IRIS can foster participation and offered code changes, as well as support the growth and branching of our code base by independent parties, through a nice range of identity management, security settings, and asynchronous communication channels such as pull requests and issues.

The challenge with moving internally maintained software to the community is coming to terms with accepting some loss of control. The code is exposed for all to see and visitors can clone and fork projects and create their own derivatives. There is a risk of branding confusion in the case of such proliferation and an uncertainty about the authenticity of a tool offered by one developer that has the same name as another. While this is something that the DOE pilot should consider, IRIS has deemed this to not be a notable threat to its own developed products in practice.

Another challenge with community software is having too many developers being able to change things without peer review. This is mitigated in repositories like GitHub by keeping privileged users to a small, designated group of principal developers that moderate all changes and establish a process of code review that is effective and timely. For larger collaborations, it can be a matter of governance to designate a separate review panel of dedicated developers that can approve a block of changes for release.

While members of the community have their own individual needs and wants when it comes to how a software tool operates, it is the community at large that have an interest in the stability of the feature sets and behavior of that software even as incremental changes are introduced. This means that broad participation in auditing the software and its latest versions prior to general availability is highly encouraged. As with many community efforts, transparency is essential.

## 5. CHALLENGES IN BUILDING A DIVERSE WORKFORCE

It is well documented that the Geosciences statistically shows poorly on its staff and faculty diversity metrics in comparison to a majority of scientific and academic fields.<sup>15</sup> IRIS, being an integral part of this community, is examining its policies and hiring practices to ensure that its organization improves on its inclusion of underrepresented groups, including women, persons of color, and members of the LGBTQIA+ community.<sup>16</sup>

For any community platform, such as a shared software repository, strong and clear up-front policies about conduct that show zero tolerance for harassment and degrading remarks, most especially directed at underrepresented individuals must be established and made a part of the registration process for any user that signs on to the platform. To enforce these policies, moderators should be assigned to not only monitor harassing behavior but have the tools to effectively and properly address situations that involve any form of discrimination and/or makes an attendee feel unwelcome or unsafe.

A related issue that has a long history in technology circles are terms and phrases that have colonial, stereotypical, or otherwise harmful connotations that can leave members feeling uneasy or even devalued. There is a growing movement to identify technical terms that should be retired from use in favor of benign terms that mean the same thing. The push by tech companies to comply with this has been met with opinions for and against.<sup>17</sup> Nonetheless, the DOE can take a strong stand for inclusion by addressing correct terminology substitutions up front in their tools, processes, and documentation. Indeed, a careful use of software branding and acronyms must also be made to follow such guidelines. Moderators can inform members about terms to avoid, working to the benefit of inclusion of their peers.

Finally, positive messaging and representation on web pages, documents, and videos in support of the DOE pilot effort should be emphasized, portraying characters and situations that represent underrepresented groups in a positive and inclusive light. A diverse workforce can only exist if all its members feel they are on the same level playing field and feel truly appreciated by their peers and supervisors without unfair judgment or derision.

## 6. REQUIREMENTS, BARRIERS, AND CHALLENGES TO TECHNOLOGY TRANSFER

As mentioned above, IRIS has struggled with the dichotomy of retaining control, branding, and trust of software we've developed versus gaining the benefit of open and collaborative work from many contributors to a project. As open

---

<sup>15</sup> <https://eos.org/articles/geosciences-make-modest-gains-but-still-struggle-with-diversity>

<sup>16</sup> [https://www.iris.edu/hq/about\\_iris/jedi\\_efforts](https://www.iris.edu/hq/about_iris/jedi_efforts)

<sup>17</sup> <https://www.wired.com/story/tech-confronts-use-labels-master-slave/>

online collaboration tools and repositories have become common and widespread, we have found that the proposition of moving to a shared and open forum of development has gained broad appeal, both for IRIS staff and for its community.

As illustrated earlier, IRIS has embraced GitHub to house many of its most recent projects to be actively worked on and where code is packaged and released. The open source code that is publicly available can be easily tracked, monitored, discussed, and contributed to. IRIS maintains control of these projects by carefully vetting who has write or maintainer permissions for each, and in some cases external core contributors are made co-maintainers to spread the load of governance of the product. The process of accepting and validating contributions from community authors has become much simpler by following the pull-request moderated merge model.<sup>18</sup>

The benefits of this model for IRIS and its community are clear: by opening the source code base for broad consumption and contribution, sustainment load is removed from limited staff time, where some packages continually place demand on its maintainers to eliminate errors and add new features. Historically, the community has always been vocal in terms of indicating what they want to see in a tool, and what they do not. By allowing the more technically inclined in the community to craft their own solutions to add to the product, precious staff time is not only spared, but desired releases tend to come to fruition in a timelier fashion. A vibrant open source community can quickly generate a robust, rich, and targeted product. All that is required is a good review and change management process to ensure that new releases are stable, backward compatible, and predictably introduced.

The barriers that are often encountered with a shared repository are disagreements on solutions, poor quality assurance processes, and slow moderator responsiveness to issues and changes. In addition, the introduction of numerous dependencies can bring a project in to a murky world of mixed licenses. Some open source licenses are extremely permissive in their terms of use, copying, and distribution, whereas others have many strings attached, such as GPL's copy-lefing of all accompanied software. Having undesirable licensing requirements for dependencies can present a difficult challenge for a shared repository, so moderators should use good license review and scanning processes to ensure that the aggregate license agreement is as open as possible, both to commercial as well as academic use. In addition, a DOE pilot shared repository should present clear up-front requirements for documented licensing on any projects it distributes, using established standards for markup that are consistent across products.

---

<sup>18</sup> <https://docs.github.com/en/pull-requests>



## 7. OVERALL SCOPE OF STEWARDSHIP EFFORT

The Scope of this effort as outlined in the RFI largely covers the important and necessary areas of concern. The one area of focus that was not evident in the RFI is the notion of FAIR compliance.<sup>19</sup> The concept of Findable, Accessible, Interoperable, and Reusable is being widely put into practice in many scientific fields, most notably for data. There is also a movement to have FAIR apply to software.<sup>20</sup> The DOE can take a leadership position in creating a FAIR repository by considering the need for:

- a. Easy search and discovery mechanisms, not only through exploratory web applications, but also through command line search tools.
- b. Simple download and build avenues through robust web service protocols that permit versioned, programmatic access to dependencies for the purpose of CI/CD in remote build environments.
- c. Redundancy and resiliency as a component of availability. The repository system should be multiply redundant and distributed so as to permit always-on, reliable access to dependencies.
- d. Releases compatible with multiple platforms, with code being pre-verified and approved for specific OS and container environments or known to work with specific virtual machine environments.
- e. Sustainment of many historical stable versions of a piece of software so that code builds can be reliably built for many years to come.
- f. Mechanisms for repositories to document persistent identifiers and references for citation. Placing emphasis on clear guidelines for citing software packages helps to reinforce the utility of the efforts of its authors as well as the value of the shared repository itself.

## 8. MANAGEMENT AND OVERSIGHT STRUCTURE

As with many initiatives that are driven by a community, the management and oversight of a shared scientific computing repository must consist of a core set of caretakers that have the mission statement of building and fostering the growth of such a repository. This group would be led by a program manager that provides vision and direction for the project. This program manager represents the director of that program and would hire and lead both line managers and technical staff to carry out the aims of that program.

To best serve the community, the caretaker group would need guidance from key stakeholders that consider different interests. These would take the form

<sup>19</sup> <https://www.go-fair.org/fair-principles/>

<sup>20</sup> Chue Hong, N. P., Katz, D. S., Barker, M., Lamprecht, A.-L., Martinez, C., Psomopoulos, F. E., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., Honeyman, T., et al. (2021). FAIR Principles for Research Software (FAIR4RS Principles). *Research Data Alliance*. DOI: [10.15497/RDA00065](https://doi.org/10.15497/RDA00065)

of committees with members elected to the post by their peers and closely models the governance structure of the IRIS Consortium.<sup>21</sup>

- a. An executive committee to oversee the business operation of the program. Budgets would be approved, and apportionment of efforts would be given clear priorities. All efforts proposed by the program manager would undergo scrutiny as to the efficacy of the business model with an eye for sustaining the program through its funding cycles.
- b. A steering committee consisting of working technical and scientific members whose charge is to indicate community requirements and desires. This group would review initiatives and efforts by the program itself and weigh to usefulness of these endeavors. New initiatives would be proposed and weighed as to their practicality and utility to the community.
- c. A coordinating committee, which is charged with ensuring that effective communication occurs between the executive and steering committee to various program arms that implement budgets and build out the capabilities of the program. Any items of confusion between oversight and guidance memoranda are discussed and disambiguated to ensure effective directives that carry the program forward.

## 9. ASSESSMENT AND CRITERIA FOR SUCCESS

It is good for an organized effort to seek metrics for tracking and indicating success, both for goal seeking in the developers and for progress reporting to stakeholders. Though IRIS does not currently adopt any form of metrics or reporting regarding its shared code repositories, we feel we can at least offer some suggestions of what good yardsticks of measure would be, generally as a rate over a specific period of time:

- a. Number of repositories
- b. Number of active repositories
- c. Number of new repositories
- d. Number of members
- e. Number of active members
- f. Number of new members
- g. Number of check-ins/merges
- h. Number of releases
- i. Number of downloads
- j. Number of issues filed
- k. Number of issues resolved/closed
- l. Number of tests run
- m. Number of tests passing
- n. Lines of code tested

---

<sup>21</sup> [https://www.iris.edu/hq/about\\_iris/governance](https://www.iris.edu/hq/about_iris/governance)

o. Number of citations

The many metrics of this sort serve to highlight the activity, vibrancy, and most of all utility of the shared repository, cementing its importance to continue and grow through generous sustainment funding.

10. FINAL THOUGHTS

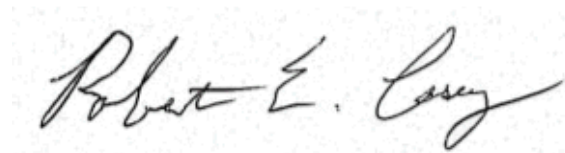
Scientific computing software, data, and compute resources have become ever more interconnected in the past decade. With broadband networking being ubiquitous in just about every point of the developed (and emerging) world, noted efforts to help bring together heterogeneous, hidden, and balkanized software package repositories into an easily accessible, trusted, and vibrant hub of activity would be of immense benefit to all technicians and researchers in the scientific community. Indeed, the great and accelerating advancements in software development and scientific discovery can be very much attributed to the ease of quickly benefiting from the work of other brilliant minds and building on top of that to create works that are new and inventive.

If the DOE were to bring such a repository into fruition, IRIS and its partners would be extremely interested in becoming both beneficiaries as well as active contributors. IRIS has a wide variety of tools that are in constant use by the geophysical community. We can only imagine how many more computational scientists would benefit if they were more widely known.

We appreciate the opportunity to offer our comments and information to the DOE's proposed program and hope you will find the concepts outlined above to be of some use as you enter the planning stages for how to go forward.

If you have questions, please feel free to reach out to IRIS Data Services and we will be happy to discuss this matter further.

Sincerely,

A handwritten signature in black ink, reading "Robert E. Casey". The signature is fluid and cursive, with the first name "Robert" and last name "Casey" clearly legible, and "E." in the middle.

Robert Casey,  
Deputy Director, Cyberinfrastructure  
IRIS Data Services  
<rob.casey@iris.edu>

