Illinois Rocstar LLC
108 Hessel Boulevard
Champaign, IL 61820-6574
+1-217-417-0885 • www.illinoisrocstar.com


Response to

Stewardship of Software for Scientific and High-Performance Computing
U.S. Department of Energy, Office of Science

Illinois Rocstar LLC
108 Hessel Boulevard
Champaign, Illinois 61820-6574

William A. Dick, President & Chief Executive Officer
Mark D. Brandyberry, Chief Technical Officer & Chief information and Security Officer
Technical Staff Contributors: Alessandro Gondolo, Akash Patel, Seth Pemberton, and Taiyo Wilson

13 December 2021

About the Respondent

Since 2007, Illinois Rocstar has been developing and providing predictive modeling and simulation tools and solutions for industry, science, defense, and security. We bring the combined strengths of engineers, computer scientists and visualization experts to produce and deliver software for cutting edge problems. Designing and building both proprietary and open-source software, Illinois Rocstar (IR) builds computational platforms for aerospace, materials, structures, and geological applications.

Founded in the legacy of the DOE NNSA/ASCI Center for Simulation of Advanced Rockets, IR is a commercially viable small business enterprise. The company has been a frequent participant in the SBIR and STTR programs and has been awarded nearly 50 Phase I and 19 Phase II/A/B projects. Our customer base includes DOE (NNSA Labs, ANL, INL, ORNL, OS), DoD, NASA and myriad US companies and universities.

We are particularly well-suited to respond to this Request for Information. Our products are custom software with flexible options for client ownership or licensing, more than half of which have been developed under DOE sponsorship targeted for DOE laboratory and/or DOE prime contractor application. Our flagship open-source *Rocstar Multiphysics* code is a product of ASCI initiatives and has been updated and augmented with ASCR funding. Subject matter expertise in IR is broadly based in multiphysics modeling and scientific simulation, including fluid flow, solid mechanics, thermal science, chemistry, geosciences and more. Maintaining our technical leadership

position has required the acquisition of deep knowledge of parallel HPC and desktop computer code development for physical and scientific systems, as well as experience packaging scientific tools in pipelines with innovative interfaces and automation. To implement and transition these engineering and software products, we have built expertise in graphical user interfaces and scientific visual display, including use of cloud-based systems such as Amazon GovCloud.

---

***Introductory Summary: DOE has released a Request for Information to explore stewardship of software for scientific and high-performance computing. They ask for information informing the discussion about how to steward the ecosystem of software that is produced by, and supports, the scientific enterprise. DOE seeks seeking input from the entire community because this issue affects the entire community. Scientific software stewardship is multi-faceted, potentially including but not limited to training and workforce support, infrastructure and curation, and shared engineering resources and project support.***

***(1) Software dependencies and requirements for scientific application development and/or research in computer science and applied mathematics relevant to DOE's mission priorities.***

Package management is a complex problem with solutions that have evolved over decades. The concern is unchanged since the 1960 and '70s, but the available tools and avenues of exchange have grown. The key question in resolving dependencies and approaches lies in two major categories: create a full ecosystem (the traditional approach, used in majority of Linux distros) or isolate each software (e.g., macOS bundles, Python virtual environments, Ubuntu Snaps).

Illinois Rocstar, as a small, high tech, computational science business, would find significant benefit in DOE guidelines on software stewardship at the initiation of a new software project. Without the benefit of enterprise level support available at much larger software companies, we often do not have institutional resources to maintain best stewardship practices. At the start of an SBIR Phase I project, when implementation of best practices is most valuable, the financial risk to survival is too high to additionally invest in infrastructure and knowledgeable personnel necessary, if those personnel can even be found and hired. Hiring skilled DevOps personnel has proven to be difficult.

*(a) What software packages and standardized languages or Application Programming Interfaces (APIs) are current or likely future dependencies for your relevant research and development activities?*

Languages/API: C, C++, Fortran, Python, MPI (various flavors), Many third-party libraries, solvers, HDF, CGNS, automated test fixtures, OneAPI (for CPU optimized matrix operations).

Software/Algorithms: Eigen, PETSc, BLAS, LAPACK, OpenFOAM, MFEM, CalculiX, TensorFlow, PyTorch, CUDA, CuDNN, SLURM or PBS for cluster management and job submission, Cmake for software and build management. Other large software systems such as MOOSE, NEAMS, Dakota, RAVEN.

*(b) What key capabilities are provided by these software packages?*

Parallel programming, data structures, solvers, I/O, CI/CD and testing.

*(c) What key capabilities, which are not already present, do you anticipate requiring within the foreseeable future?*

Automated security scanning; advanced processor support; easier builds/installation would be nice; some pre-built Docker images or App images for different operating systems.

*(d) Over what timeframe can you anticipate these requirements with high confidence?*

We see the need for these maintenance capabilities over the next 15 years, perhaps longer. Requirements for enhanced security may last far longer.

*(e) What are the most-significant foreseeable risks associated with these dependencies and what are your preferred mitigation strategies?*

The most significant risks relate to version incompatibilities: third-party library changes over time; OS changes over time; automated library scanning and conflict resolution with community tools; poor code/API documentation; very complicated API (e.g., TensorFlow API in C/C++).

***(2) Practices related to the security and integrity of software and data:***

*What strategies and technology do you employ, or intend to employ in the foreseeable future, to ensure the security and integrity of your software and its associated provenance metadata? What capabilities do you provide, or intend to provide in the foreseeable future, to assist users of your software with ensuring scientific reproducibility, recording the provenance of their work products, securing their information, protecting the privacy of others, and maintaining the integrity of their results?*

All of this falls under our Simulation Lifecycle Management initiatives, of which IR has several. Currently we do not build SLM into the physics software, but are developing external job, input/output, and provenance capture tools. Have not yet considered building SLM directly into the physics software.

For machine learning enabled products, we have training data that we need to protect. This training data can be generated in-house or provided by the client.

It would be useful to the community for DOE to facilitate secure, cloud-based software delivery systems. There are secure file transfer protocols such as MFT (Managed File Transfer). National labs are increasingly using that but it's not something central.

***(3) Infrastructure requirements for software development for scientific and high-performance computing:***

*What infrastructure requirements do you have to productively develop state-of-the-art software for scientific and high-performance computing? These requirements might include access to testbed hardware, testing allocations on larger-scale resources, hosting for source-code repositories, documentation, and other collaboration tools. What are the key capabilities provided by this infrastructure that enables it to meet your needs? What key capabilities, which are not already present, do you anticipate requiring within the foreseeable future? Over what timeframe can you anticipate these requirements with high confidence? What are the most-significant foreseeable risks associated with this infrastructure and what are your preferred mitigation strategies? When responding to these questions, please describe the scope of the relevant research and development activities motivating the response.*

We have a cluster in-house but also can access NCSA resources. More routine access to some of the National lab clusters would help ensure adequate infrastructure.

Small business risk includes having too much compute power sitting idle (CPU/GPU) unbalanced by having too little, or none, due to some unforeseen outages. Presently for machine learning, we have a GPU machine, though as we implement more ML-based projects and products, we will need access to something off-site/in-house with more GPU compute capability.

Source repository (local Gitlab), CI/CD infrastructure (dedicated hardware, GitLab runner facilities), Automated code documentation from comments (Doxygen, other), access to large core-count test machines (1000 core+), automated static code analysis (Lint), tools for packaging code on multiple operating systems. Automated deployment systems.

Need, but don't have robust parallel code analysis/debugging runtime tools. They exist but are expensive.

***4) Developing and maintaining community software:***

*How much additional effort is needed to develop and maintain software packages for use by the wider community above the effort needed to develop and maintain software packages solely for use in specific research projects or for internal use? What tasks are the largest contributors to that additional effort? What are the largest non-monetary impediments to performing this additional*

*work? How is any such additional effort currently funded? How does that funding compare to a level of funding needed to maximize impact?*

We have very limited experience in this area. We have been trying to nurture communities around some of our software (ACFD, Promesh, Rocstar). For now, we are using Github public repos for software delivery to a broader community. In the future, we can envision having some packages, pre-built docker images, appimages, etc. for the community users.

Deployment on public systems (security). Answering questions from non-specialists. Providing public Q&A platforms. Producing software with build systems that are robust enough to support a range of OS targets. Effort: significant.

Developing internal codes vs. those to be used by an open-source community/external users certainly present different challenges. Most immediately, it is difficult to ensure our software will run on different operating systems and different versions of various software with unique dependencies. This requires more extensive testing on our part and remains the biggest challenge to us (and likely to all software producers).

***(5) Challenges in building a diverse workforce and maintaining an inclusive professional environment:***

*What challenges do you face in recruiting and retaining talented professionals to develop software for scientific and high-performance computing? What additional challenges exist in recruiting and retaining talented professionals from groups historically underrepresented in STEM and/or individuals from underserved communities? What challenges exist in maintaining inclusivity and equity in the development community for scientific and high performance computing software? What successful strategies have you employed to help overcome these challenges? What opportunities for professional recognition and career advancement exist for those engaged in developing scientific and high-performance-computing software?*

We get very limited applications and since we have very specific needs for hiring due to the nature of our projects, the pool of applicants gets even more scarce. I think we have done a good job in the diversity of people that we have here based on experience, background, etc. Difficult to find qualified engineers and scientists with CSE talent. Also difficult to attract them to fly-over country. We get few applications from underserved groups but have been successful in hiring a few from these groups.

For diversity of field experience in the user community, we are unable to extract diversity data related to who is using software downloaded from online GitHub repos. We anticipate establishing training sessions and other avenues to give these users an incentive to identify themselves so that we can get ourselves introduced to them.

We maintain relationships with relevant academic/industrial institutions, such as the University of Illinois (our local Research I University), in particular universities with Computational Science and Engineering (CSE) programs, and NCSA. We are members of several organizations relating to scientific and engineering computing (AIAA, ANS, APS, NAFEMS, etc.) and present/publish our work in M&S sections in conferences/journals.

***(6) Requirements, barriers, and challenges to technology transfer, and building communities around software projects, including forming consortia and other non-profit organizations:***

*ASCR recognizes that successful software for scientific and high-performance computing often has many stakeholders, including academic research activities, research laboratories, and industry. Moreover, while DOE has provided funding for the development of a significant number of foundational software packages within the modern software ecosystem for scientific and high-performance computing, as the complexity of the software ecosystem continues to increase, and number of stakeholders has grown, ASCR seeks to understand how it might encourage sustainable, resilient, and diversified funding and development models for the already-successful software within the ecosystem. Such models include, depending on circumstances that ASCR seeks to better understand, both the private sector and non-profit organizations. Non-profit organizations include both charitable organizations (e.g., those with 501(c)(3) status) and R&D consortia (e.g., those with 501(c)(6) status). What are the important characteristics and components of sustainable models for software for scientific and high-performance computing? What are key obstacles, impediments, or bottlenecks to the establishment and success of these models? What development practices and other factors tend to facilitate successful establishment of these models?*

We have been successful in building relationships with national labs, research institutions, and universities. IR has been part of several successful STTR projects over the past 15 years. Being part of an R&D consortium is beneficial to small businesses because they can contribute based on their niche expertise into a larger effort and assimilate the wisdom of the team. That said, strong project management, strict software/code standards, and precise end goal in mind are very important for consortia. The average university faculty member is not, in general, interested in effective project management.

The key obstacle for building a community around software is marketing the product based on diverse needs of users in industries. This is especially true for capabilities we provide that are not easily available in commercial software or are very expensive compared to our solution. Trust is another issue. People in industry won't readily trust or use open-source solutions because of lack of benchmarking and polished user experience.

Engagement is the primary impediment. On most public code repository platforms, thousands of downloads may occur with no knowledge of who the downloaders were, and with no way to engage with them directly. Very few ever contact us. Difficult to build a community around a niche piece of multiphysics software that way.

We have executed several SBIR projects to mature DOE-sponsored software for industrial use. These involve things like code hardening, enhancing and modernizing dependencies and libraries, developing graphical/web interfaces for usability, HPC deployment, and modernizing major code bases with state-of-the-art physics solvers. DOE could do a much better job in advertising the products of its/our consortia and identifying users within the DOE community. In short, "visibility" of the software needs o be enhanced.

***(7) Overall scope of the stewardship effort:***

*The section labeled Potential Scope, mentioned earlier in the RFI, outlines activities that ASCR currently anticipates potentially including in future programs stewarding the software ecosystem for scientific and high-performance computing. Are there activities that should be added to, or removed from, this list? Are there specific requirements that should be associated with any of these activities to ensure their success and maximize their impact?*

How much do we want DOE to manage these efforts? Maybe oversight can be handled by DOE but specific roles can be assigned to national labs/private institutions.

Stewardship implies maintenance. Some of the hardest things about using outside software is "library rot" where old software library versions are required, and absent those libraries, the software does not run on newer OS versions, along with build system incompatibility. For instance, trying to meld a modern Cmake-based code with an older makefile-based library can be a nightmare. Providing updating to codes for newer library and OS versions should be a DOE goal, and it likely must be automated. Somehow standardizing build systems would also be useful but is likely more difficult. If a stewardship organization is stood up that has control over this type of thing, a system should be devised to run tests on the software in the repository on new OS versions and post the results along with the software (assuming that the software isn't under active development by others). This will thereby identify what software will not be able to be used on newer OS versions (without work).

Much HPC-capable software comes out of the national labs. However, due to security concerns, the labs are often multiple OS-versions behind what commercial use of such software requires on their own machines. Whether using ASCR offerings in commercial tools, or selling commercial tools back into the labs, multiple OS version support is important.

Potential Scope: Scientific software stewardship is multi-faceted, potentially including, but not limited to:

• Training: Providing training on software-development best practices and the use of core software.

• Workforce support: Providing outreach and support activities to build and maintain a diverse, skilled workforce with opportunities for professional recognition and career advancement.

• Infrastructure: Providing infrastructure for software packaging, hosting, testing, and other common capabilities.

• Curation: Establishing governance processes and standards to enable resource allocation in the most-effective manner balancing stability with the need to satisfy evolving requirements.

• Maintaining situational awareness: Defining, publishing, and communicating understandable information about relevant software and its dependencies; collecting information from users and deployment requirements from facilities.

• Shared engineering resources: Providing software-engineering resources to assist with maintenance activities of key projects, including triaging problems from testing and adjusting for new compilers; system-software and platform versions; and changing package requirements.

• Project support: Providing support for the continued development of key projects, including enhancing them to function efficiently on new hardware platforms; take advantage of emerging hardware and software technologies; comply with best practices; and otherwise provide high priority features desired by other users.

***(8) Management and oversight structure of the stewardship effort:***

*What do you anticipate will be effective models for management and oversight of the scientific and high-performance-computing software ecosystem, and how would that management structure most-effectively interact with DOE and other stakeholders? In addition to DOE, who are the key stakeholders? How can the management structure coordinate with DOE user facilities and others to provide access to relevant testbed systems and other necessary infrastructure?*

Tasking non-dedicated government employees to host the stewardship effort will likely not be useful if they have other jobs to do. More likely to succeed is to task a private organization to provide dedicated resources to organize, publish, maintain, and "advertise" the software. They should be tasked with collecting reference to all DOE ASCR software in one web-accessible place (some might be links to other places, but mature software that is not being actively developed should be centralized). Doing the OS testing noted in #7 should be part of their effort, potentially using DOE systems to minimize cost. Monthly or quarterly meetings among the organization and DOE ASCR stakeholders would be especially useful. Web-based interaction with non-DOE stakeholders, including a web-forum for questions, suggestions, requests that is monitored daily. They could be the funnel for users to access DOE compute resource requests. This organization could also assemble other DOE non-ASCR software in the same place to provide a "one-stop-shop" for DOE software. Potentially provision of compiled versions of software for multiple platforms (this is labor intensive unless using CI/CD and automated deployments in a mature DevOps environment).

***(9) Assessment and criteria for success for the stewardship effort:***

*What kinds of metrics or criteria would be useful in measuring the success of software stewardship efforts in scientific and high-performance computing and its impact on your scientific fields or industries?*

The principal criterion of success is that it is easy for the user to find needed software that works on their system. This will require that curated software remain current with newer OS versions, or at least flagged with current incompatibilities. This may demand automated library upgrades; easy to ask (and get answers to) general questions; monitoring and enforced testing.

Success metrics may include (i) the amount of collaboration increment between academia and industry; (ii) number of projects undertaken/managed under this effort; (iii) number of successful outcomes (commercial applications, software for general benefit; (iv) improvement in productivity in certain processes; (v) some measure of significant impact in daily lives of the public, etc. Apart from items (ii) and (iii), there is no obvious objective/numerical assessment aligned with these "metrics." That said, these are important outcomes.

***(10) Other:***

*What are key obstacles, impediments, or bottlenecks to progress by, and success of, future development of software for scientific and high-performance computing? Are there other factors, issues, or opportunities, not addressed by the questions above, which should be considered in the context of stewardship of the ecosystem of software for scientific and high-performance computing?*

A key obstacle — maybe the key obstacle — in modeling and simulation software is availability of trusted examples, use cases, and verification and validation cases for end-use packages that can be accessed and used for testing, learning, and prototyping new cases. Full test suites would be nice, though are rare in practice. Taking an ASCR code and modifying it should have robust regression tests performed to show that the base functionality remains capable, but the full test suites are often not available.