Cover Page

Category: State of the Profession

Advocating for Equality of Contribution: The Research Software Engineer (RSE)

Rebecca Ringuette[1,2], Nicholas Murphy[3], Maksym Petrenko[2], Kevin Reardon[4,5], Josh Rigler[6], Leila Mays[2], Silvina Guidoni[2,7], Darren De Zeeuw[2,8], Robert Weigel[9], Thomas Y. Chen[10], Mike Liemohn[11], Ryan Timmons[12], Yihua Zheng[2], Alexa Halford[2], and Jeff Klenzing[2].
[1]ADNET Systems Inc, 6720B Rockledge Dr., Suite 504, Bethesda, MD 20817, USA,
[2]NASA Goddard Space Flight Center, Greenbelt, MD 20769, USA
[3]Center for Astrophysics | Harvard & Smithsonian
[4]National Solar Observatory, Boulder, CO, 80303, USA
[5]University of Colorado, Boulder, CO 80303, USA
[6]USGS, Geomagnetism Program, Golden, CO 80225, USA
[7]American University, Washington, DC 20016, USA
[8]Catholic University of America, Washington, DC 20064, USA
[9]George Mason University, Fairfax, VA 22030, USA
[10]Columbia University, New York, NY 10027, USA
[11]University of Michigan, Ann Arbor, MI 48109, USA
[12]USGS

Co-signers:
Lutz Rastaetter[2], Sam Schonfeld[13], and Micah Weberg[9].
[2]NASA Goddard Space Flight Center, Greenbelt, MD 20769, USA
[13]Institute for Science Research, Boston College, Newton, MA 02459, USA
[9]George Mason University, Fairfax, VA 22030, USA

Synopsis: (limit of 400 characters)

Heliophysics depends on RSEs to properly engineer software. However, RSEs receive unequal treatment compared to their science counterparts, resulting in unsustainable talent loss. These restrictions include lack of credit for their contributions and insufficient training. This paper describes what a RSE is and proposes solutions, including implementing appropriate recognition standards.

Current Landscape:
Today's scientific software in space weather and Heliophysics is predominantly developed and maintained by scientists who have learned to code and software engineers that have learned science, whether by formal or informal training (e.g. the Van Allen Probes Science Gateway, and open source code packages: Burrell et al 2018 and Angelopoulos et al. 2019). Producing such stable, quality software for public scientific use requires a complex skill set — that of a scientist combined with a software engineer (SE). A person with this complex skill set is known as a Research Software Engineer (RSE). While there are shining examples of well-developed software packages that have become pillars of the community, there are many more examples of codes produced for a single or series of publications. Transitioning such codes into public use takes a significant amount of effort, which unfortunately is not properly recognized or funded by the scientific community and leadership at various organizational levels. Those that do perform this task, typically one or more RSEs, spend an appreciable amount of time to gain and refine the necessary skills.

For those that become skilled in this area, the lack of key elements such as reputational growth, proper training, and attractive career options preclude any resemblance of success in space weather and Heliophysics, resulting in a constant flow of talent from our field into the software industry. Realistically, some of the flow of software talent to the industry is driven by salary differentials, with little that can be done to resolve it. However, if the field does not strengthen its offerings of proper training, enhanced recognition, intellectual stimulation, and more relaxed and congenial workplace culture, we will never be able to compete.

In some cases, this flow of talent benefits our efforts by igniting collaborations between the software industry and government and academic efforts. However, this outflow of talent more typically results in significant setbacks in our technological growth as we are left with the task of training or recruiting new talent. Thus, a new batch of graduate/post-doctoral students are acquired for software development, who then become skilled and leave the field for more promising careers with better salaries and faster promotion cycles. **This constant cycle of training and depletion in skilled scientific software developers and engineers is a waste of our resources.** We must focus on retaining the talent we train and more efficiently training those who are interested. Our efforts as currently directed are not sustainable in the long term, and warrant immediate attention in the face of our growing technological demands.

In practice, we have observed four loosely defined categories of people involved in software development for research purposes.

- Scientists who put in the effort to apply best practices to code development.

- SEs who work to find a middle ground between applying the most modern standards and meeting specific science goals.
- Scientists who focus on publication-oriented code or single use code for the specific analysis within the paper without applying best practices.
- SEs who focus on implementing the most updated standards and technology at the expense of the science goal.

Codes produced by the first pair of categories tend to be more stable and more easily usable by others in the community. In comparison, codes developed by the second pair of categories tend to either be inoperable on other machines or too costly to maintain due to the constant application of changing standards. There are multiple examples of software developed by members of the community in the first two categories, which are in direct contrast with the many more examples of code from the second pair of categories. In some cases, code created can and should be for single-use applications, as opposed to incorporation into a software package, as long as the code is documented and developed for reproducibility and for others to build upon (Gil et al. 2016). However, projects are often not given the proper support to produce higher quality software, resulting in many years of effort spent on code that could have a larger impact on the community. Typically, the problem reduces to open source software development not being funded or contributing to metrics which enable career progression (e.g. tenure review). As a result, the field becomes overwhelmed with packages of limited scope and stability, further compounding the issues of open science related to software.

Definition of a RSE:

We define Research Software Engineers (RSEs) as individuals that are trained as scientists, typically with a PhD in a science field, or an individual trained as a software engineer, whose primary product is software that advances science (as opposed to papers that describe advances in science). The US Research Software Engineer Association (US-RSE) defines RSEs as "those who regularly use expertise in programming to advance research." This definition broadly encompasses scientists who program, software engineers who work on research software, and everyone in-between[1]. This is distinct from a Software Engineer (SE), whose training is in computer science or a related field and whose primary product is not focused in a science field. Although their responsibilities may overlap in some cases, their typical primary tasks are distinct. A RSE's primary responsibility is to develop code for scientific applications, such as modeling code or data pipeline software, possibly also applying GPU or containerization techniques. In contrast, a SE's primary responsibility is to design and maintain complex software systems, such as a network of containerized modeling codes in an automated continuous integration continuous deployment (CI/CD) pipeline or a rotating data storage

[1] https://us-rse.org/about/what-is-an-rse/

system located on the cloud. These classifications should not be confused with hardware engineers, who focus on developing and maintaining computational hardware systems.

A Path Forward:

While there have been PhD scientists who specialize in code development for decades, there is now more and more fundable work that requires effective computing skills. This growing demand is reflected in other sectors as well. Simply put, there is more to the tactful execution of a software project than there ever has been before, which requires the skills of an RSE. **We therefore advocate for a new, more recognized career track in space weather and Heliophysics for Research Software Engineers.**

RSEs have the challenge of developing code using the application of best practices in software development while still addressing the science goals of the project in a timely fashion. Balancing these requirements well requires training that is similarly balanced. Considering the varying backgrounds of typical RSEs, we envision two typical training pathways for RSEs: one to transition a scientist to an RSE and another to transition a SE to an RSE. Exposing scientists to the topics described below while still a student is one additional method to increase RSE talent inflow. This could be accomplished either through existing short courses and summer schools or by allowing a course from those described to be counted as a math or computer science elective.

The first scenario would begin with a science degree, preferably a PhD, in a chosen science field of interest. Afterwards, the scientist may spend some time in research before beginning the transition or may immediately decide to transition. Once a scientist decides to acquire advanced software skills, they should take a series of classes. The non-science background needed for a RSE to become successful include modern coding practices, parallelization techniques, GPU and CPU acceleration, CI/CD pipelines, open-sourcing code, writing tutorials and documentation, software marketing, and the basics of containerization. This series of classes should include basic applications of each topic in science software development and should result in a graduate level or professional certification or master's degree in scientific software design. **We advocate for a degree of this structure to be created at various institutions, preferably offered online for additional flexibility**, and beyond the tools and resources currently available[2]. Combined with the science background, this certification/degree provides the complete skill set necessary for an RSE to successfully develop software that is applicable to a large range of science cases, usable and understandable by the community, and maintainable through an open-source CI/CD pipeline such as GitHub.

[2] https://merely-useful.tech/py-rse/, https://urssi.us/winterschool/

Portions of this proposed degree program already exist at various institutions with more in development, such as a selection of courses led by Fernando Perez at UC Berkeley[3], the materials in preparation by the NASA TOPS effort[4], and a variety of courses in the computer science and software engineering degree programs at multiple institutions. We recommend that such a degree program be based upon these and similar options, and note that it will take time to develop. In the meantime, a graduate or professional certificate program could be developed based on test implementations of the concepts in institution-based training courses. The certificate program could be offered online through existing avenues such as MIT and the University of Colorado.

Alternatively, a RSE may begin as a SE and show interest in acquiring a deeper understanding of a given science topic. That person's transition to a RSE position would require either a master's or PhD degree, in the chosen science field. Typical graduate level projects would include the application of their software training to solve a problem in science, similar to those tackled in the scientific software design program described above. Regardless of the pathway chosen, the additional training for an SE or scientist to transition to become a RSE should be allowed to occur while in the position with the permission of the hiring entity.

Regardless of training, RSEs will not remain in the space weather and Heliophysics fields unless their work is recognized as a significant contribution to research. The components of this issue are intertwined, but each is connected to recognition for software development. At the most basic level, quality contributions to software development need to be perceived in a manner similar to a science publication. Guidance should be given to institutions to weigh software publications and contributions on equal standing with more traditional science publications. In addition to the typical software functionality checks, this review process could also include feedback from the team the RSE is a part of, and various qualities of the software developed (e.g. capability, science outcome and impact, and documentation quality). Additionally, software citation standards in traditional publications need to be adopted by the community to further recognize usage of software (Niemeyer et al. 2021).

Software packages and libraries need to be subject to a peer-review process – such as already implemented for the Journal of Open Source Software (JOSS) and being discussed by the Python in Heliophysics Community[5] – to improve the reproducibility and usability of open-source tools that can be used by the wider community. Such reviews should include best

[3] https://bids.berkeley.edu/people/fernando-p%C3%A9rez, https://o365coloradoedu-my.sharepoint.com/:v:/g/personal/juba8233_colorado_edu/EWMbbJ8ywkVPtygLYDZaAcYBFD2uZusUucNDRGidtIeY1Q?e=z840iZ
[4] https://github.com/nasa/Transform-to-Open-Science
[5] https://heliopython.org/

software practices, such as code documentation, CI/CD deployment, multi-platform support. Single-use codes should typically accompany scientific papers with sufficient documentation and best practices applied - but may not be readily deployable on other platforms. However, this clean and well documented code should be sufficient to reproduce results with other coding languages. When possible, these codes should be built on top of community packages and libraries to minimize the parts that are single use.

The peer-review issue becomes more complex when the contribution does not result in a publication, such as maintenance for software packages, contributions to larger packages or modeling codes, or infrastructure advancements. One way to more properly attribute credit for these scenarios is to provide an avenue for peer-reviewed short publications on these and similar advances, such as provided in JOSS when applicable. Another approach is to simply review the contributions of RSEs in a way similar to that of an SE, such as via software releases[6]. Other methods include incorporating software usage metrics and a code review process separate from publication avenues. A combination of these approaches will likely be the best solution. Correcting the current bias in performance reviews will significantly affect the careers of RSEs and SEs by pushing the culture towards equality and thus increasing our talent retention and enabling more scientific innovation.

A recent executable paper demonstrates the **efficiency possible when scientists, RSEs and SEs work together**. Polson et al. (2022) created a science workflow combining multiple python packages in an online containerized environment for scientists to implement in a complex model-data comparison effort in magnetospheric physics. Without the RSEs and SEs on the team, the scientist would have required a much longer time investment to be able to accomplish the same work. By including the RSEs and SEs on the team, the science goal was achieved with much less effort overall. This collaboration is a primary motivation for funding agencies to **include RSEs (and SEs) pay in budgets and proposals**. Similar collaborations between RSEs, SEs and scientists will be absolutely necessary to properly address the more complex publication requirements of truly open science.

Recommendations:

This white paper has presented a multi-pronged approach for training and retaining talent in research software engineering. This approach is built upon the following facets:

- **Develop and implement recognition standards** for RSEs appropriate to their contributions as described.

[6] https://www.usgs.gov/products/software/software-management

- **Connect potential RSEs with programs in software engineering and science** to more effectively train scientists who are interested in software development, and to train software engineers that are interested in science. These training programs should result in transferable skills that advance science.
- **Develop graduate level certificate and masters programs** for scientists to become educated in the basics of best software engineering and development practices.
- Offer **additional funding opportunities** to transition codes developed with public funds to be made open-sourced and be transitioned for public use using software engineering best practices. This requirement should be implemented in a method compliant with the National Academy of Sciences recommendation: “Any open source software policy that NASA Science Mission Directorate develops should not impose an undue burden on researchers; therefore, any policy should be as simple as possible and any mandates should be fully funded." (NAS 2018)
- **Educate scientists** on what RSEs can do, and demonstrate how those capabilities accelerate research progress.
- Supply the additional funds needed for individual proposals, missions, and data archives to **offer salaries and benefits** appropriate for the important contributions provided by RSEs.

We must take action to provide the proper training, attribute the proper credit, and reduce the flow of RSE talent from Heliophysics. Unless the problems outlined here are addressed promptly, our field will continue its unsustainable downward spiral in efficiency, costing the community large portions of our time and resources. We cannot afford to allow this troubling trend to continue. **We must act now.**

The example set by the executable paper referenced above is a promising one, and can be modeled by other science teams. The combined efforts of scientists, RSEs and SEs with a united purpose, such as demonstrated in the creation of the executable paper, are a powerful factory of science advances aligned with the goals of FAIR and open science (Wilkinson et al. 2016). The field of Heliophysics is ready and waiting for the application of modern technology to our science. We need to remove the barriers preventing these powerful collaborations from occurring with greater frequency, and prepare for the exciting results to come from these teams. The sooner our field is made more equal for RSEs, the sooner our field can advance.

References:

Angelopoulos, V., P. Cruce, A. Drozdov, E. W. Grimes, N. Hatzigeorgiu et al. (2019). The Space Physics Environment Data Analysis System (SPEDAS). *Space Sci. Rev.*, **215**, 1, 9. https://www.doi.org/10.1007/s11214-018-0576-4.

Burrell, A. G., A. Halford, J. Klenzing, R. A. Stoneback, S. K. Morley, et al. (2018). Snakes on a spaceship—An overview of Python in heliophysics. *Journal of Geophysical Research: Space Physics*, 123, 10,384–10,402. https://doi.org/10.1029/2018JA025877.

Gil, Y., C. H. David, I. Demir, B. T. Essawy, R. W. Fulweiler, et al. (2016). Toward the Geoscience Paper of the Future: Best practices for documenting and sharing research from data to software to provenance, *Earth and Space Science*, 3, 388–415, https://doi.org/10.1002/2015EA000136.

National Academies of Sciences, Engineering, and Medicine. 2018. An Open Source Software Policy for NASA Space Science. *Washington, DC: The National Academies Press*. https://doi.org/10.17226/25217.

Niemeyer, K. E., A. M. Smith, and D. S. Katz (2021). The Challenge and Promise of Software Citation for Credit, Identification, Discovery, and Reuse. *Journal of Data and Information Quality*, **7**, 4, https://doi.org/10.1145/2968452.

Polson, S., R. Ringuette, L. Rastaetter, E. Grimes, J. Niehof, N. A. Murphy, and Y. Zheng (2022). Making an Executable Paper with the Python in Heliophysics Community to Foster Open Science and Improve Reproducibility. Submitted to *Frontiers in Astronomy and Space Sciences.* *https://deepnote.com/workspace/shawn-polson-c095a0fb-f02d-416d-9c94-c4a9c4e8e54d/project/PyHC-Paper-EBuWRj_QSXikjqTz5wiqrA/%2FPyHC%20Tech%20Paper-rewrite.ipynb*

Science gateway: Overview. *Van Allen Probes Science Gateway* Available at: https://rbspgway.jhuapl.edu/. (Accessed: 29th July 2022)

Wilkinson, M., M. Dumontier, I. Aalbersberg, G. Appleton, M. Axton, et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* 3, 160018. https://doi.org/10.1038/sdata.2016.18.

What is an RSE? *US-RSE* (2022). Accessed at https://us-rse.org/about/what-is-an-rse/.