# Reusability in Science: From Initial User Engagement to Dissemination of Results

Ketan Maheshwari*, David Kelly*, Scott J. Krieder†, Justin M. Wozniak*,
Daniel S. Katz‡, Mei Zhi-Gang§, Mainak Mookherjee¶
*MCS Division, Argonne National Laboratory
†Department of Computer Science, Illinois Institute of Technology
‡Computation Institute, University of Chicago & Argonne National Laboratory
§Nuclear Engineering Division, Argonne National Laboratory
¶Department of Earth and Atmospheric Sciences, Cornell University



***Abstract*—Effective use of parallel and distributed computing in science depends upon multiple interdependent entities and activities that form an ecosystem. Active engagement between application users and technology catalysts is a crucial activity that forms an integral part of this ecosystem. Technology catalysts play a crucial role benefiting communities beyond a single user group. An effective user-engagement, use and reuse of tools and techniques has a broad impact on software sustainability. From our experience, we sketch a life-cycle for user-engagement activity in scientific computational environment and posit that application level reusability promotes software sustainability. We describe our experience in engaging two user groups from different scientific domains reusing a common software and configuration on different computational infrastructures.**

***Index Terms*—Technology-catalyst, user-engagement, scientific computation**

## I. Introduction

Domain scientists often have limited time to investigate the capabilities that a large scale computing and data-handling infrastructure combined with a high performance software framework could bring to their scientific activities. Technology catalysts help speed up the tedious process of organizing scientific computations such that they can be easily mapped onto computational infrastructure. However, this is an iterative process and not free of pitfalls. The source of these pitfalls can be the scientific process itself or a mismatch in technical requirements mapping to computational infrastructures.

This presents a challenge: enabling effective reuse of existing user-engagement patterns and related products for a new scientific user. If this challenge can be met, it could lead to a considerable acceleration in the process of conceptualizing, describing, defining, deploying, and executing scientific experiments. Reuse of data and software libraries is fairly common. Reuse of enabled applications across scientific domains is not as common. A successful execution of such applications might require tuning specific to the application requirements. However, with familiarization from previous engagements, much of this process can be expedited.

A widely reused system has a higher sustainability as a community supports its maintenance. Enabling application level reuse promotes sustainability of the entire ecosystem of modern science. In this experience paper, we report on the following:

1) Experience in scientific community engagement describing activities performed at different levels in order to support scientific users with applications deployed onto new, larger and faster systems.
2) A sketch and demonstration the elements of a successful scientific application deployment cycle.
3) Enhancements of an enabling software framework based on user feedback resulting in a software with improved usability.

The remainder of this paper is organized as follows. In Section II, we describe the user engagement cycle that provides a context to the human aspects of our work. In Section III, we describe the applications on which our experience is based and in which we apply software reuse techniques. In Section IV, we describe the hardware and software complexities that make software maintenance strategies important. In Section VI, we summarize some related work, and in Section VII we offer concluding remarks.

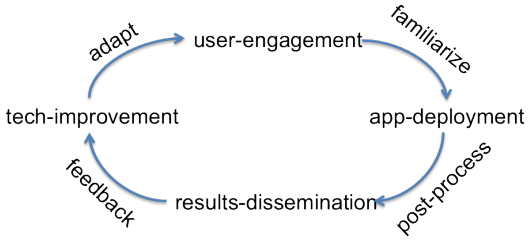


Fig. 1. Activities and transitions in user engagement cycle.

## II. User Engagement Cycle

User engagement with technical catalysts is a complex social process that differs with respect to institutions, culture, and technical practices. A distillation of key points in the process is diagrammed in Figure Fig. 1. The cycle consists of four phases and transition activities between these phases. It starts with user-catalyst communication involving familiarizing the domain science and technology from the

user side and computational tools and technology from the catalyst side with each other. The result is a map of scientific tools onto the computational infrastructure. The next phase is application deployment and execution on these resources. Enabling software tools are put into practice in this phase. This ends in post-processing activity involving collection, pruning, structuring of results. The next phase is the results-dissemination phase. During this phase, the science results obtained in the previous activity are presented to the scientific user. Adaptive users often do further analysis themselves and trigger a next stage via their feedback. The feedback results in technological improvements and adjustments to better suite the requirements of application on hand.

In our experience, demonstrating the results of engagement with one user-group to the other has significantly inspired, and aroused interests in adapting modern high performance computational techniques. This method led to a productive cycle of defining and executing experiments which has benefited both groups. This affirms the role of technology catalysts by enabling the use of cross-application knowledge.

From a software sustainability perspective, identifying software patterns and applying them into new applications is an opportunity to maximize the initial investment in the software research, as well as speed progress in later projects. An example of this process, based on application experience, is described in the next section.

## III. User Groups

In this section we describe the two user groups we engaged with in this work. First, the Mineral Physics Group at the Department of Earth and Atmospheric Science at Cornell University, and second, the Material Science Group at Argonne National Laboratory. As described below, the application areas differ greatly (geological research vs. nuclear energy), yet the underlying software tools and challenges have much to gain from software reuse.

### A. Mineral Physics

Rocks and minerals exposed at the surface of the Earth are in constant interaction with the overlying hydrosphere. Over geological time scales, this leads to stabilization of hydrous mineral phases such as serpentine [1]. These lock up oceanic water. These minerals are then dragged down along subduction zones. This process has a long term influence on the global sea levels. In order to understand how much water is dragged into the deep interior of the Earth, one must have a better understanding of the energetics and thermo-elasticity of these hydrous phases stable at subduction zone conditions.

We use ab-initio simulations based on density functional theory to understand the energetics and thermo-elasticity of hydrous mineral phases relevant to the subduction zone conditions. The mineral phases stable at the subduction zone conditions often have lower space-group symmetry and a large number of atoms in their unit cell. The compute requirements of simulations are directly proportional to the number of atoms in a unit cell of a mineral. Consequently the simulations are computationally intensive.

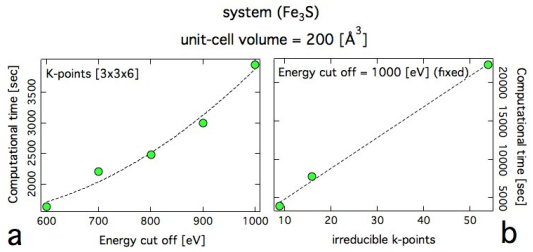


Fig. 2. Constant volume $Fe_3S$ unit-cell experiments: (a) Plot of computational time as a function of cut-off energy for a constant k-point mesh of 3x3x6; (b) plot of computational time vs. irreducible k-points for a constant cut-off energy.

For our research we use the Vienna Ab-initio Simulation Package (VASP) [2] software suite. VASP has been quite successful in predicting band structure, ground state energy, and physical properties including elasticity. From our experience using institutional resources, we find that a simulation for a system-size of approximately 100 atoms requires 24 hours of CPU time. For a single mineral phase, we typically need 40-50 simulations to determine the full elastic constant tensor and its pressure dependence. One comprehensive analysis on a mineral phase requires a combination of parameters to test for convergence of ground state energy. This requires intensive use of computational resources. In addition, we explore solid-solution thermodynamics across various end-member chemistry of the mineral phases.

The initial, relatively small study with 32 atoms was conducted in two stages. First, a determination of volume at which the energy is minimum was conducted. From this, the unit cell volume of 200 angstroms was chosen for further analysis over a constant 3-D volumetric mesh (Fig. 2-a) and constant energy (Fig. 2-b). Fig. 2-b shows that over five hours of normalized CPU time is required for a single cell volume analysis.

### B. Material Science

Cerium dioxide ($CeO_2$) is an important material with a wide range of technological applications. It is used as an electrolyte in solid oxide fuel cells (SOFCs), as a catalytic converter in the automotive industry, and as a model material for $PuO_2$ in nuclear energy applications. It has a fluorite structure and can develop a complex pattern of defects, depending on temperature and oxygen pressure [3]. In order to fully understand and to be able to accurately predict the micro-structural evolution under irradiation, elucidating the underlying formation mechanisms of the extended defects is important.

The purpose of this study is to provide a unified view of the effect of non-stoichiometry and temperature on the formation and evolution of nano-scaled defects, i.e., defect clusters, voids and fission gas bubbles, in irradiated Ceria using existing experimental data, multiscale modeling, and computer simulations.

The migration barriers calculations using the DFT-NEB (Density Functional Theory-Nudged Elastic Band) method and the defect structure evolution calculations using molecular dynamics are computationally expensive. The computations require a two-stage interdependent execution of VASP over the material structure definition. The first stage performs relaxation and static calculations to predict atomic positions. The second stage DFT-NEB calculations rely on the structures produced by the previous stage. A few hundred thousand CPU hours are expected for the calculations. Currently, we are using high performance computing clusters with more than 2,000 cores connected by fast InfiniBand.

### C. Computational Tools and Techniques

While the two applications belong to different scientific domains, their computational profile is similar. Both use the VASP software tool with similar input data structures and computational stages. Both applications first run a calibration stage to find a close structural range before moving to a final compute intensive stage. Consequently, the general computational structure of the applications is similar, and they can be run using the same computational tools, specifically the Swift [4] parallel scripting tool. Because a full technical description of Swift is out of scope for this paper, the following points summarize Swift's capabilities.

- Swift is an open-source, Apache-licensed software framework for distributed computing designed for scientific users. A C-like scripted language expresses applications as workflows.
- Swift makes it easy to run ordinary application programs on parallel and remote computers (from laptops to supercomputers).
- Swift works with a variety of resource managers and file transfer protocols. It uses ssh, PBS, Condor, SLURM, LSF, SGE, Cobalt, and Globus to run applications, and scp, http, ftp, and GridFTP to move data.
- Swift's `foreach` statement is the main workhorse of the language; it executes all iterations of a loop concurrently. The actual number of tasks executed in parallel is based on available resources and settable "throttles".

Workflows in the form of scripts result in portable and flexible expressions of applications. For instance, running a single study using different simulation/software packages from different vendors or calculation approaches becomes convenient without changing the application logic. Additionally, portable expression of workflows enables the use of multiple computational infrastructures, as described in the next section.

## IV. Cyberinfrastructures and Technology

In both of our application cases, users were introduced to new computational infrastructures, in our case the NSF-funded XSEDE and the Argonne Laboratory Computing Resource Center (LCRC) Blues systems. The minor adaptation of highly portable application Swift scripts resulted in minimal application porting delays before making use of these systems, demonstrating software sustainability in the scripts and other technological investments.

In the rest of this section we describe the enhancements in Swift triggered directly or indirectly as a result of the user engagement activities.

### A. Configuration

With a complex set of communication and execution methods on local and remote systems, Swift offers a highly sophisticated, fine tuned to-the-core configuration options to users. However, the disadvantage of this that users often have to adjust parameters in different configuration files. To get around these inconveniences, a unified and abstracted approach to configuration was designed. Under this approach, a single file, structured as name-value pairs, is employed to record diverse properties such as execution sites, data management, application management and miscellaneous configuration options (e.g. retry counts). Additionally, predefined templates with a spectrum of configurations help users select one (out of the box) that works for their requirements.

### B. Galaxy

While a textual representation of application and a terminal based execution like in Swift gives users flexibility and more control, often a visual interface carries more appeal to users. With this requirement in mind, we have conceptualized an integration of Swift with the Galaxy portal environment [5]. The integration offers users an interactive, visual interface to large scale computational systems while benefiting from both Swift's and Galaxy's strengths in the scientific community. With this development, independent Swift applications can be turned into Galaxy executable tools and used from within the Galaxy environment. A user dialog enables users to enter application and high-level execution parameters such as the target execution site on which to run the application.

### C. Accelerators

Special hardware systems such as accelerators are gaining prominence in HPC systems. The host CPU offloads work to an accelerator such as a graphics processing unit (GPU) which relieves the CPU of precious compute cycles. However, code reusability on these systems is essentially non-existent, resulting in customized porting issues of codes written using specific programming languages that are capable of targeting accelerators (e.g., CUDA, OpenCL, OpenACC). Additionally, there are performance, portability, and programmability issues arising from conflicting architectures (e.g., NVIDIA vs. AMD vs. Intel Xeon Phi). One solution so far has been modular code, with clearly defined inputs and outputs. VASP and related software [6] have been extended to work with NVIDIA GPUs through use of CUDA [7]. Since computation is dependent on architecture, codes have to be maintained for all of these architectures, reducing portability. GeMTC (GPU Enabled Many-Task Computing) addresses these issues from a Many-Task Computing approach, by developing a library of GPU kernels that are callable from Swift. The effort targets the lowest

level of hardware and is working towards a maintainable and runnable architecture-agnostic high-level software tool [8].

## V. Software reuse

In this section, we show a snapshot of Swift and wrapper scripts used and reused in the two engagements. The code below shows Swift's 'app' definition for VASP call.

```
app (file o, file e, file outcar, file contcar)
run_vasp (file vasp_incar, file vasp_poscar,
          file vasp_potcar, file vasp_kpoints,
          string _dir, string _subdir)
{
  runvasp @vasp_incar @vasp_poscar
          @vasp_potcar @vasp_kpoints
          _dir _subdir stdout=@o stderr=@e;
}
```

**Swift binding to VASP program**

The following Swift code snippet is used for mineral physics application.

```
string dirs[] = [ "300", "400", "500", ... ];
file out[]<simple_mapper;
           location="stdouts",
           prefix="vasp", suffix=".out">;
foreach dir, i in dirs {
  file incar<dir,"/","INCAR">;
  file kpoints<dir,"/","KPOINTS">;
  file potcar<dir,"/","POTCAR">;
  file poscar<dir,"/","POSCAR">;
  out[i] = vasp(incar, kpoints, potcar, poscar);
}
```

**VASP usage by minerals physics application**

The following Swift code snippet is used for materials science applications respectively.

```
string dirs[] = [ "0L", "1L1", "1L3", ...];
string subdirs[] = ["pos0","pos1"];

foreach dir in dirs {
 foreach subdir in subdirs {

/* file declarations omitted
for brevity */

  (output, error, outcar, contcar) =
    runvasp(incar_relax, poscar, potcar,
            kpoints, dir, subdir);
 }}
```

**VASP usage by materials science application**

The following code shows the common wrapper script used.

```
#!/bin/bash

incar=$1
poscar=$2
potcar=$3
kpoints=$4
dirname=$5
subdirname=$6

mpiexec -machinefile $PBS_NODEFILE \
  /blues/nfs/home/jlow/vasp/vasp.5.3/vasp

cp OUTCAR  output/vasp-outcar-$dirname.$subdirname
cp CONTCAR output/vasp-contcar-$dirname.$subdirname
```

**Reusable shell script to launch parallel VASP under Swift**

## VI. Related Work

The challenges and value of developing robust software for scientific computing has been documented in the past [9]. The idea of workflows as enablers of reusable software is well embedded in scientific community. The subject of reusable and sustainable tools has been addressed by the Taverna [10] and Cactus [11] developments. A library of scientific workflow components, acting as a virtual shelf of applications, has been conceived in the past. Pegasus [12] is a framework for mapping complex scientific workflows onto distributed systems. Pegasus combined with HUBzero[13], an online platform for dissemination and collaboration, provides a mechanism for enabling the masses to utilize scientific workflows [14]. RunMyCode.org [15] is cloud-based tool to expedite the process of reproducing results. Researchers share their software with collaborators and reviewers who can easily test and evaluate the work without concern for the underlying hardware or environment.

## VII. Conclusion

In this paper, we presented our experience of engagement between technology catalysts and scientific users. We describe the process of application-level reusability via a typical case wherein the commonalities between two distinct applications benefited the respective user communities. Additionally, we described how Swift is a simple and effective tool for task parallel computing on high-performance and/or distributed systems that offers unique capabilities to promote software sustainability.

Two aspects of the experience in this work helped improve the Swift framework: identifying usage patterns and science user feedback. We expect the Swift usability enhancements will enable wider community adoptation and improve ease of conducting science on multiple infrastructures. We expect such a collaborative science to have short and long term benefits by inspiring similar efforts in the broader community across disciplines. Our experience demonstrates that effective reusability of software in science is crucial and involves much more than just the technical aspects.

## VIII. Acknowledgments

This work was partially supported by the U.S. Department of Energy, under Contract No. DE-AC02-06CH11357. Some work by DSK was supported by the National Science Foundation, while working at the Foundation. Any opinion, finding, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

[1] M. Mookherjee and L. Stixrude, "Structure and elasticity of serpentine at high-pressure," *Earth and Planetary Science Letters*, vol. 279, no. 1-2, pp. 11–19, 2009.

[2] G. Kresse and J. Furthmüller, "Software VASP, vienna (1999)," *Phys. Rev. B*, vol. 54, no. 11, p. 169, 1996.

[3] D. S. Aidhy, D. Wolf, and A. El-Azab, "Comparison of point-defect clustering in irradiated $CeO_2$ and $UO_2$: A unified view from molecular dynamics simulations and experiments," *Scripta Materialia*, vol. 65, no. 10, pp. 867 – 870, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1359646211004507

[4] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster, "Swift: A language for distributed parallel scripting," *Par. Comp.*, vol. 37, pp. 633–652, 2011.

[5] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor *et al.*, "Galaxy: A platform for interactive large-scale genome analysis," *Genome research*, vol. 15, no. 10, pp. 1451–1455, 2005.

[6] F. Spiga and I. Girotto, "phiGEMM: A CPU-GPU library for porting Quantum ESPRESSO on hybrid systems," in *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on*. IEEE, 2012, pp. 368–375.

[7] M. Hutchinson and M. Widom, "VASP on a GPU: Application to exact-exchange calculations of the stability of elemental boron," *Computer Physics Communications*, vol. 183, no. 7, pp. 1422–1426, 2012.

[8] S. J. Krieder and I. Raicu, "Towards the support for many-task computing on many-core computing platforms," Doctoral Showcase, IEEE/ACM Supercomputing/SC, 2012.

[9] D. A. Aruliah, C. T. Brown, N. P. C. Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. Huff, I. Mitchell, M. Plumbley, B. Waugh, E. P. White, G. Wilson, and P. Wilson, "Best practices for scientific computing," *CoRR*, vol. abs/1210.0530, 2012.

[10] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat *et al.*, "Taverna: A tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, 2004.

[11] G. Allen, T. Goodale, F. Lffler, D. Rideout, E. Schnetter, and E. L. Seidel, "Component specification in the Cactus Framework: The Cactus Configuration Language," *CoRR*, vol. abs/1009.1341, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1009.html

[12] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[13] M. McLennan and R. Kennell, "HUBzero: A platform for dissemination and collaboration in computational science and engineering," *Computing in Science & Engineering*, vol. 12, no. 2, pp. 48–53, 2010.

[14] M. McLennan, S. Clark, E. Deelman, M. Rynge, K. Vahi, F. McKenna, D. Kearney, and C. Song, "Bringing scientific workflow to the masses via Pegasus and HUBzero," *parameters*, vol. 13, p. 14.

[15] V. Stodden, C. Hurlin, and C. Perignon, "RunMyCode.org: A novel dissemination and collaboration platform for executing published computational results," in *E-Science (e-Science), 2012 IEEE 8th International Conference on*, 2012, pp. 1–8.