# How to Professionally Develop Reusable Scientific Software—And When Not To

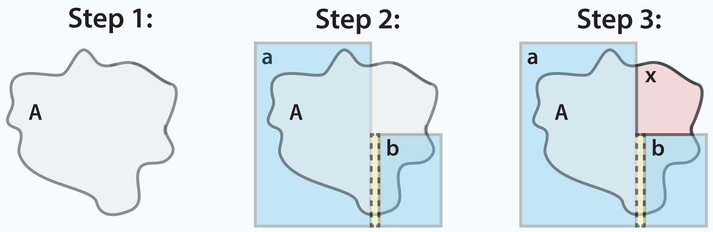
Vyas Ramasubramani[1], Carl S. Adorf[1], Joshua A. Anderson[1], Sharon C. Glotzer[123]

1. Department of Chemical Engineering 2. Department of Materials Science and Engineering
3. Biointerfaces Institute, University of Michigan, Ann Arbor, MI 48109

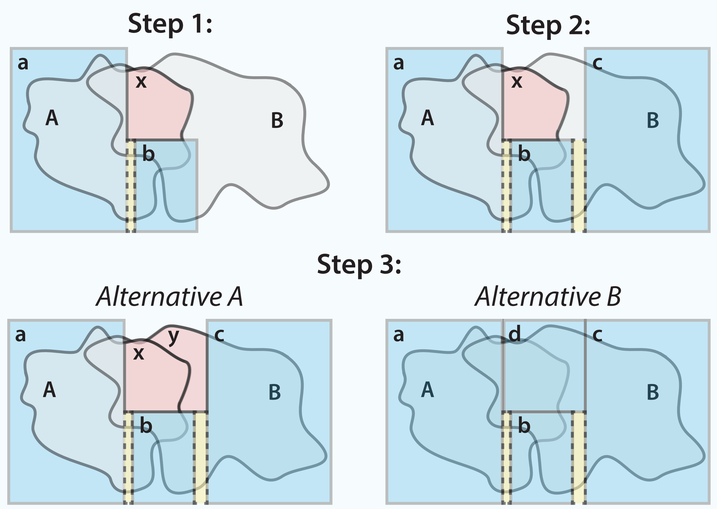
## Developing Computational Solutions

In theory, computational projects would be perfectly scoped from the start, allowing exact determination of what existing tools could be used and the reuse potential of new developments. In practice, project scopes change constantly, making it extremely difficult to accurately assess which code will be reused.

### Timeline: Problem A




### Timeline: Problem B




Solving a computational problem (grey) involves:

- finding existing well-defined packages (blue)
- implementing missing pieces with one-off prototype code (red)
- writing adapters (yellow) to interface between packages and protoypes

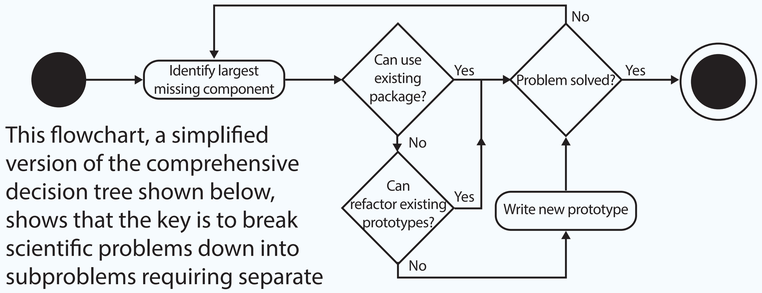
There are two options when existing prototypes solve part of the problem:

- implementing new protoypes to fill the holes (Alternative A)
- refactoring existing prototype into a package (Alternative B)

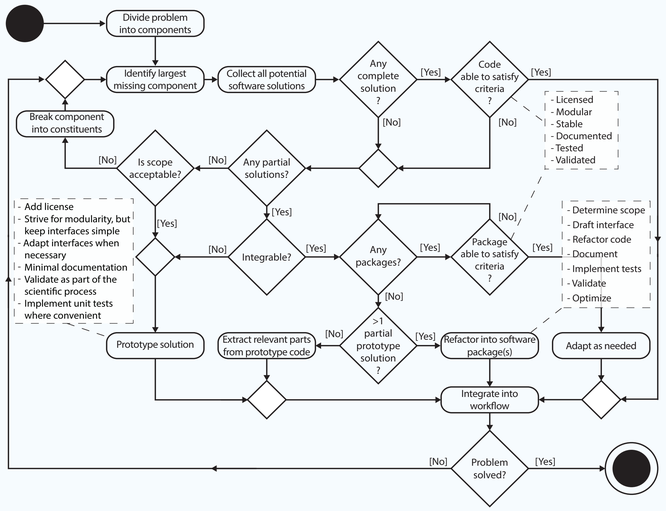
Determining when to choose each alternative and how to proceed is the crux of our lazy refactoring approach.

## Applying Lazy Refactoring

The lazy refactoring approach involves constantly reevaluating existing software whenever a new computational problem arises.




This flowchart, a simplified version of the comprehensive decision tree shown below, shows that the key is to break scientific problems down into subproblems requiring separate software solutions. These are analyzed to determine whether existing code, preferably in package form, can be used to solve them. New code, when needed, is always a prototype.



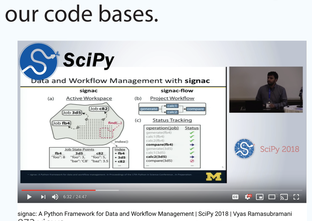

## Community Development

By presenting our open-source software in various forums, we

- attract a broad base of users and contributors
- help integrate our software with the broader ecosystem
- improve our software stack's usability and sustainability

We also foster group involvement in development by, for instance, holding periodic hackathons to upgrade and maintain our code bases.



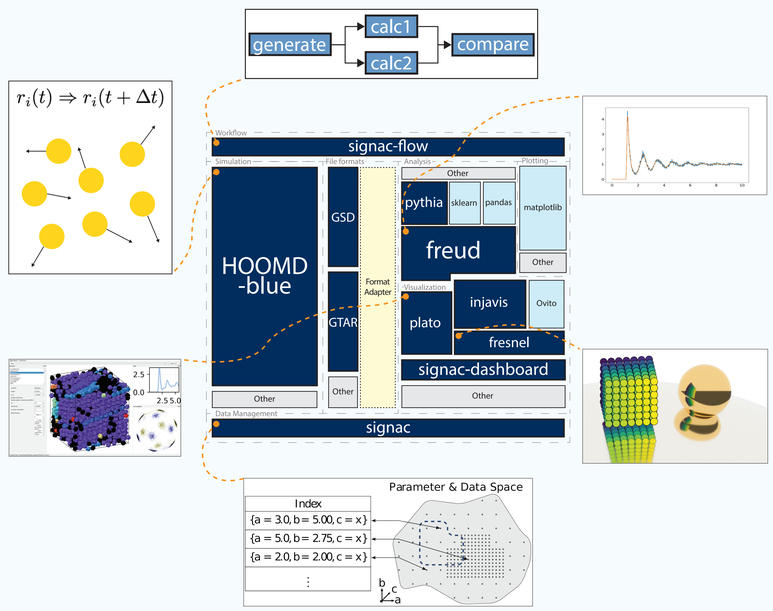



## Lazy Refactoring Over Time

Lazy refactoring has led to clear interfaces between well-defined groups of software tools that we developed in-house (navy blue):

- Molecular simulation: HOOMD-blue
- Simulation analysis: freud, pythia
- Simulation visualization: plato, injavis, fresnel
- Data and workflow management: signac, signac-flow, signac-dashboard
- Utilities: GSD, GTAR

These tools also interoperate with well-known components of the broader scientific Python ecosystem (light blue).




In addition to creating and maintaining these code bases, we also provide various other services to facilitate the broader usage of our code, such as:

- project websites including information on how to cite our code
- online documentation hosted through services like Read The Docs
- distribution through package managers like pip and conda
- support chat rooms (like Gitter) and mailing lists (like Google Groups)

Providing such services increases our visibility, and consequently the likelihood that others can reuse our work to accelerate scientific progress.

## Acknowledgements

We would like to thank all contributors to the group's code base, including all members of the Glotzer group as well as the numerous external contributors to our software listed on the respective websites. We would like to thank Michael Engel for his development of injavis, some of which was done while in the Glotzer group. The software toolkits, packages and methodologies described in this paper have been supported both internally (injavis and plato) and by a number of grants, principally: DOD/ASD(R&E) under Award No. N00244-09-1-0062, "Smart, Autonomous. Adaptive Phenomena in Self-Organizing, Reconfigurable Materials," (2009-2014) for HOOMD-blue v0.7 – v1.1, freud and later support for injavis; National Science Foundation, Award # DMR 1409620, "CDS&E: Fast, scalable GPU-enabled software for predictive materials design & discovery," (2014-2018) for HOOMD-blue development, especially DEM and HPMC, and for fresnel, as well as project conceptualization and early implementation for signac; and MICCoM, as part of the Computational Materials Sciences Program funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division, under Subcontract No. 6F-30844 for development and deployment of signac.

[1] C. S. Adorf, V. Ramasubramani, J. A. Anderson and S. C. Glotzer, "How to professionally develop reusable scientific software — and when not to,". Computing in Science & Engineering (2019). doi: 10.1109/MCSE.2018.2882355