# Digging Deeper Into the State of the Practice for Domain Specific Research Software

**Spencer Smith**, Peter Michalski

Computing and Software Department
Faculty of Engineering
McMaster University

ICCS SE4Science, June 23, 2022

# Outline

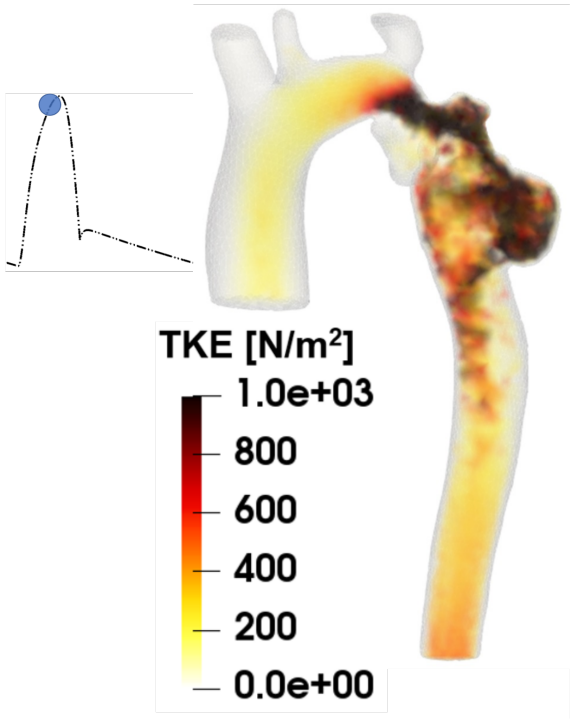
- Motivation and Example
- Overview of SOP Assessment Methodology
- Identify Software (RQ1)
- Rank
  - By Best Practices (RQ2)
  - Versus Community Ranking (RQ3)
- Comparison to Other Research Software
  - Artifacts (RQ4)
  - Tools (RQ5)
  - Processes (RQ6)
  - Pain Points (RQ7, RQ8)
- Threats to Validity
- Concluding Remarks

# Motivation

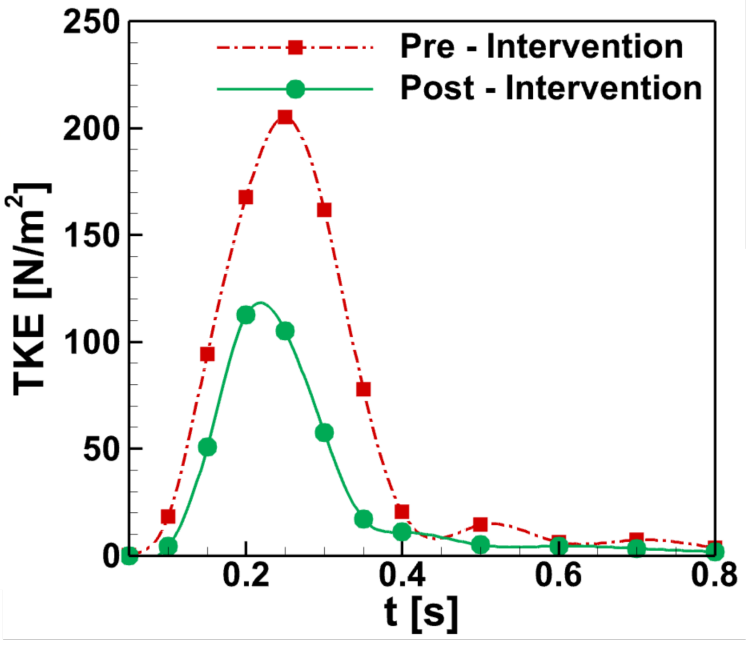
- Previous studies:
  - Survey developers (Hannay et al., 2009; Nguyen-Hoan et al., 2010; Pinto et al., 2018).
  - Mining (Grannan et al., 2020; Sood et al., 2019).
  - Recruit broadly, or by prog lang (Pinto et al., 2018), or role of developer (Nangia and Katz, 2017).
  - Case studies (Carver et al., 2007; Segal, 2005).
- We focus on:
  - Contents of repositories (manual and automated),
  - Interviews with developers,
  - One domain at a time.
- Include Domain Expert in the assessment.
- Help users select software.
- Learn what works in a domain, and what doesn't.

# LBM Running Example

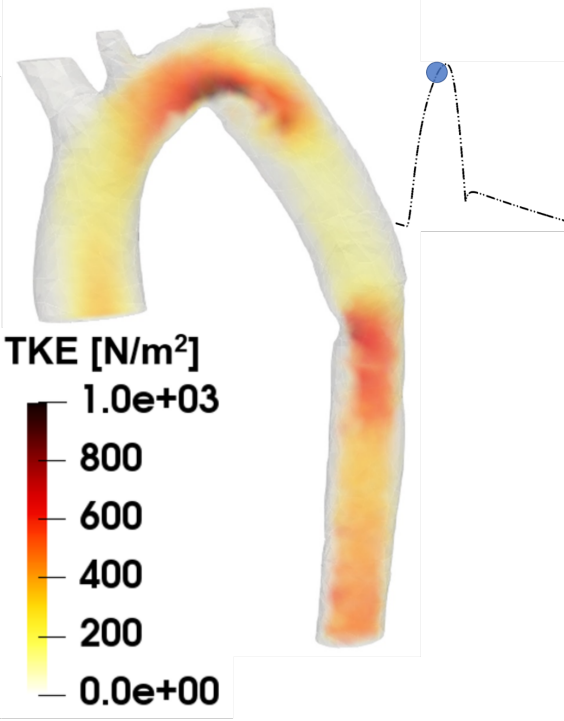
**Turbulent Kinetic Energy (TKE)**




**Volumetric integration of turbulent kinetic energy as a function of time**




**Turbulent Kinetic Energy (TKE)**



# Methodology

From Smith et al. (2021):

1. Identify the domain of interest.
2. List candidate software packages for the domain.
3. Filter the software package list.
4. Measure using the measurement template.
   - 10 qualities (installability, correctness, reliability, etc.).
   - 108 measures.
5. Use AHP to rank the software packages.
6. Interview the developers (Semi-structured).
   - 20 questions.
   - How they organize their projects?
   - Their understanding of the users?
   - Current and past difficulties?
   - Solutions the team has found or will try?
   - Use of documentation?
7. Domain analysis.

# Identify Software (RQ1)

Chosen domain should have the following properties:

1. Well-defined and stable theoretical underpinnings.
2. A community of people studying it.
3. Open source options.
4. Approximately 30 candidate packages.

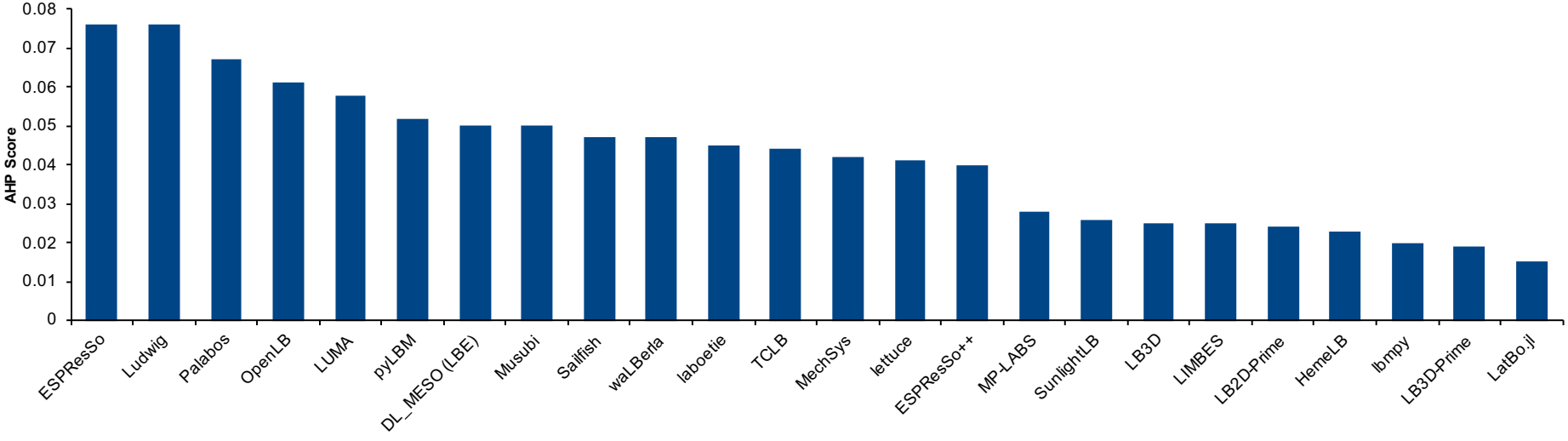
Find candidates via:

1. Search engine queries.
2. GitHub.
3. swMATH.
4. Scholarly articles.
5. Domain Expert.

| Name | Dim | Pll | Com | Rflx | MFl | Turb | CGE |
|---|---|---|---|---|---|---|---|
| DL_MESO | 2, 3 | MPI/OMP | Y | Y | Y | Y | Y |
| ESPResSo | 1, 2, 3 | CUDA/MPI | Y | Y | Y | Y | Y |
| ESPResSo++ | 1, 2, 3 | MPI | Y | Y | Y | Y | Y |
| HemeLB | 3 | MPI | Y | Y | Y | Y | Y |
| laboetie | 2, 3 | MPI | Y | Y | Y | Y | Y |
| LatBo.jl | 2, 3 | - | Y | Y | Y | N | Y |
| LB2D-Prime | 2 | MPI | Y | Y | Y | Y | Y |
| LB3D | 3 | MPI | N | Y | Y | Y | Y |
| ... | | | | | | | |
| Sailfish | 2, 3 | CUDA | Y | Y | Y | Y | Y |
| SunlightLB | 3 | - | Y | Y | N | N | Y |
| TCLB | 2, 3 | CUDA/MPI | Y | Y | Y | Y | Y |
| waLBerla | 2, 3 | MPI | Y | Y | Y | Y | Y |

| **Summary Information** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Software name? | DL_MESO | SunlightLB | MP-LABS | LIMBES | LB3D-Prime | LB2D-Prime | laboetie | Musubi |
| Number of developers | unclear | 2 | 1 | unclear | 1 | 1 | 2 | unknown |
| License? | terms of use | GNU GPL | GNU GPL | GNU GPL | unclear | unclear | GNU GPL | BSD |
| Platforms? | Windows, OS X, Linux | Linux | Linux | Unix | Windows, Linux | Windows, Linux | Linux | Windows, OS X, Linux |
| Software Category? | private | public | public | public | public | public | public | public |
| Development model? | freeware | open source | freeware | freeware | freeware | freeware | unclear | freeware |
| Programming language(s)? | FORTRAN, C++, Java | C, Perl, Python | FORTRAN, Markdown | FORTRAN | C | C, Shell | FORTRAN, Wolfram Markdown | Fortran |
| … | … | … | … | … | … | … | … | … |
| **Installability** | | | | | | | | |
| Installation instructions? | yes | yes | yes | yes | yes | yes | yes | yes |
| Instructions in one place? | yes | yes | yes | yes | yes | yes | yes | yes |
| Linear instructions? | yes | yes | yes | yes | yes | yes | yes | yes |
| Installation automated? | yes, makefile | yes, makefile | yes, makefile | yes, makefile | yes, makefile | yes, makefile | yes, makefile | yes |
| Descriptive error messages? | yes | yes | no | n/a | n/a | no | n/a | n/a |
| Number of steps to install? | 8 | 6 | 6 | 4 | 2 | 4 | 4 | 10 |
| Numbe extra packages? | 4 | 4 | 3 | 1 | 2 | 2 | 2 | 5 |
| Package versions listed? | yes | no | no | no | no | no | no | no |
| Problems with uninstall? | unavail | unavail | unavail | unavail | unavail | unavail | unavail | unavail |
| … | … | … | … | … | … | … | … | … |
| Overall impression (1..10)? | 9 | 7 | 6 | 8 | 7 | 5 | 7 | 8 |
| … | … | … | … | … | … | … | … | … |
| **Surface Reliability** | | | | | | | | |
| … | … | … | … | … | … | … | … | … |

# Ranking By Best Practices (RQ2)




67% of packages rank in top five for at least one quality

| Name | Our Rank | Stars | Star Rank | Watches | Watch Rank |
|---|---|---|---|---|---|
| ESPResSo | 1 | 145 | 2 | 19 | 2 |
| Ludwig | 2 | 27 | 8 | 6 | 7 |
| Palabos | 3 | 34 | 6 | GitLab | GitLab |
| OpenLB | 4 | N/A | N/A | N/A | N/A |
| LUMA | 5 | 33 | 7 | 12 | 4 |
| pyLBM | 6 | 95 | 3 | 10 | 5 |
| DL_MESO | 7 | N/A | N/A | N/A | N/A |
| Musubi | 8 | N/A | N/A | N/A | N/A |
| Sailfish | 9 | 186 | 1 | 41 | 1 |
| ... | ... | ... | ... | ... | ... |
| LatBo.jl | 24 | 17 | 10 | 8 | 6 |

# Compare Artifacts to Recommendations (RQ4)

- United States Geological Survey (USGS, 2019)
- DLR Software Guidelines (Schlauch et al., 2018)
- Scottish COVID-19 Consortium (Brett et al., 2021)
- Good Enough Practices in Scientific Computing (Wilson et al., 2016)
- xSDK Community Package Policies (Smith et al., 2018)
- Trilinos Developer's Guide (Heroux et al., 2008)
- EURISE Network Technical Reference (Thiel, 2020)
- CLARIAH Task Force (van Gompel et al., 2016)
- Software Quality Assurance Baseline Criteria (Orviz et al., 2017)

| Artifact | Count | LBM |
|---|---|---|
| LICENSE | 8 | C |
| README | 7 | C |
| CONTRIBUTING | 7 | U |
| CITATION | 3 | U |
| CHANGELOG | 4 | U |
| INSTALL | 4 | C |
| Uninstall | 1 | R |
| Dep. List | 3 | C |
| Authors | 3 | C |
| Code Conduct | 1 | – |
| Acknowledge | 3 | R |
| Style Guide | 4 | R |
| Release Info. | 3 | U |
| Prod. Roadmap | 3 | R |

| Artifact | Count | LBM |
|---|---|---|
| Getting started | 4 | C |
| User manual | 2 | U |
| Tutorials | 1 | C |
| FAQ | 3 | R |
| Issue Track | 6 | C |
| Version Control | 8 | C |
| Build Scripts | 6 | C |
| Requirements | 3 | – |
| Design Doc. | 6 | U |
| API Doc. | 4 | R |
| Test Plan | 2 | U |
| Test Cases | 8 | U |

# Tools (RQ5)

- Summarize tools
  - Visible in repos
  - Mentioned in interviews
- Compare to other research software
  - Version Control
    - Almost all software guides recommend
    - 81% of rsch soft (AlNoamany and Borghi, 2018)
    - 67% of LBM software use
  - Continuous Integration
    - Many software guides recommend
    - 70% of popular GitHub projects (Hilton et al., 2016)
    - 12.5% of LBM software use
    - 17% of medical image analysis (Dong, 2021)
- Observed LBM tools include code editors, verification tools, build tools, domain specific libraries, collaboration tools, document generation tools, etc.

# Processes (RQ6)

- Literature suggests
  - Agile philosophy (Carver et al., 2007; Segal, 2005)
  - Amethododical process (Kelly, 2013)
  - Peer review (Heroux et al., 2008; Orviz et al., 2017; USGS, 2019)
- LBM
  - Interviews confirm agile-like
  - ESPResSO uses peer review

# Pain Points (RQ7, RQ8)

Potential pain points (Wiese et al., 2019; Pinto et al., 2018; Katerbow and Feulner, 2018):

- lack of time,
- lack of funding,
- cross-platform compatibility,
- scope bloat,
- lack of user feedback,
- dependency management,
- reproducibility, and
- oracle problem.

Highlights for LBM:

- lack of time,
- lack of funding, and
- difficulty with ensuring correctness.

# SOP for LBM

- Developers are addressing pain points by
  - designing for change,
  - circumventing the oracle problem, and
  - prioritizing documentation and usability.
- For future improvements we suggest:
  - employing linters,
  - conducting rigorous peer reviews,
  - writing and submitting more papers on software,
  - growing the number of contributors by following current recommendations for open source projects, and
  - augmenting the theory manuals to include more requirements specification relevant information.

# Threats to Validity

- Reliability
  - One person measures all packages
  - Measurements take several months to complete
- Construct Validity
  - Qualities measured indirectly
  - Assume high ratio of comments improves maintainability
  - AHP ranking assumes equal weight between qualities
  - Approximate popularity by stars and watches
- Internal validity
  - Not all activities will leave a trace in the repos
  - Small sample of developers
- External validity
  - Generalization depends on LBM software development being similar to the development of research software in general

# Concluding Remarks

- Measuring a domain is time-consuming
  - Requires manual work for some steps
  - Approx 173 person hours per domain
- Benefits
  - Looking at what developers are doing, not just what they say they are doing
  - Help users select software
  - Customized advice for a given domain
  - Lessons and warnings for other domains
- Future Work
  - Deeper measures of usability, performance
  - Meta-analysis

# References I

Yasmin AlNoamany and John A. Borghi. Towards computational reproducibility: researcher perspectives on the use and sharing of software. *PeerJ Computer Science*, 4(e163):1–25, September 2018.

Alys Brett, James Cook, Peter Fox, Ian Hinder, John Nonweiler, Richard Reeve, and Robert Turner. Scottish covid-19 response consortium. https://github.com/ScottishCovidResponse/modelling-software-checklist/blob/main/software-checklist.md, August 2021.

# References II

Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post. Software development environments for scientific and engineering software: A series of case studies. In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*, pages 550–559, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2828-7. doi: http://dx.doi.org/10.1109/ICSE.2007.77.

Ao Dong. Assessing the state of the practice for medical imaging software. Master's thesis, McMaster University, Hamilton, ON, Canada, September 2021.

# References III

A Grannan, K Sood, B Norris, and A Dubey. Understanding the landscape of scientific software used on high-performance computing platforms. *The International Journal of High Performance Computing Applications*, 34(4):465–477, 2020. doi: 10.1177/1094342019899451.

Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. How do scientists develop and use scientific software? In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, SECSE '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-3737-5. doi: 10.1109/SECSE.2009.5069155.

# References IV

Michael A. Heroux, James M. Bieman, and Robert T. Heaphy. Trilinos developers guide part II: ASC softwar quality engineering practices version 2.0. https://faculty.csbsju.edu/mheroux/fall2012_csci330/TrilinosDevGuide2.pdf, April 2008.

Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. Usage, costs, and benefits of continuous integration in open-source projects. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 426–437, 2016.

Matthias Katerbow and Georg Feulner. Recommendations on the development, use and provision of Research Software, March 2018. URL https://doi.org/10.5281/zenodo.1172988.

# References V

Diane Kelly. Industrial scientific software: A set of interviews on software development. In *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '13, pages 299–310, Riverton, NJ, USA, 2013. IBM Corp. URL http://dl.acm.org/citation.cfm?id=2555523.2555555.

Udit Nangia and Daniel S. Katz. Track 1 Paper: Surveying the U.S. National Postdoctoral Association Regarding Software Use and Training in Research. pages 1–6. Zenodo, June 2017. doi: 10.5281/zenodo.814220.

# References VI

Luke Nguyen-Hoan, Shayne Flint, and Ramesh Sankaranarayana. A survey of scientific software development. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 12:1–12:10, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0039-1. doi: 10.1145/1852786.1852802.

Pablo Orviz, Álvaro López García, Doina Cristina Duma, Giacinto Donvito, Mario David, and Jorge Gomes. A set of common software quality assurance baseline criteria for research projects, 2017.

# References VII

Gustavo Pinto, Igor Wiese, and Luis Felipe Dias. How do scientists develop and use scientific software? an external replication. In *Proceedings of 25th IEEE International Conference on Software Analysis, Evolution and Reengineering*, pages 582–591, February 2018. doi: 10.1109/SANER.2018.8330263.

Reza Sadeghi, Nadav Gasner, Seyedvahid Khodaei, Julio Garcia, and Zahra Keshavarz-Motamed. Impact of mixed valvular disease on coarctation hemodynamics using patient-specific lumped parameter and lattice boltzmann modeling. *International Journal of Mechanical Sciences*, 217, 2022. doi: https://doi.org/10.1016/j.ijmecsci.2021.107038. URL https://www.sciencedirect.com/science/article/pii/S0020740321007244.

# References VIII

Tobias Schlauch, Michael Meinel, and Carina Haupt. DLR software engineering guidelines, August 2018.

Judith Segal. When software engineers met research scientists: A case study. *Empirical Software Engineering*, 10(4):517–536, October 2005. ISSN 1382-3256. doi: 10.1007/s10664-005-3865-y.

Barry Smith, Roscoe Bartlett, and xSDK Developers. xsdk community package policies, Dec 2018.

W. Spencer Smith, Jacques Carette, Peter Michalski, Ao Dong, and Oluwaseun Owojaiye. Methodology for assessing the state of the practice for domain X. https://arxiv.org/abs/2110.11575, October 2021.

# References IX

Kanika Sood, Anshu Dubey, boyana norris, and Lois Curfman McInnes. Repository-Analysis of Open-source and Scientific Software Development Projects. 2 2019. doi: 10.6084/m9.figshare.7772894.v2.

Carsten Thiel. EURISE network technical reference. https://technical-reference.readthedocs.io/en/latest/, 2020.

USGS. USGS software plannning checklist. https://www.usgs.gov/media/files/usgs-software-planning-checklist, December 2019.

# References X

Maarten van Gompel, Jauco Noordzij, Reinier de Valk, and Andrea Scharnhorst. Guidelines for software quality, CLARIAH task force 54.100. https://github.com/CLARIAH/software-quality-guidelines/blob/master/softwareguidelines.pdf, September 2016.

I. S. Wiese, I. Polato, and G. Pinto. Naming the pain in developing scientific software. *IEEE Software*, pages 1–1, 2019. ISSN 0740-7459. doi: 10.1109/MS.2019.2899838.

Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal. Good enough practices in scientific computing. *CoRR*, abs/1609.00037, 2016. URL http://arxiv.org/abs/1609.00037.

# Image Credits (in order of appearance)

- Are You Digging Deep? When Antivirus Is Not Enough
- Sadeghi et al. (2022)