Reproducible Science and Deep Software Variability

Mathieu Acher
IRISA, Université de Rennes 1
Institut Universitaire de France (IUF)
Rennes, France
mathieu.acher@irisa.fr

ABSTRACT

Biology, medicine, physics, astrophysics, chemistry: all these scientific domains need to process large amount of data with more and more complex software systems. For achieving reproducible science, there are several challenges ahead involving multi-disciplinary collaboration and socio-technical innovation with software at the center of the problem. Despite the availability of data and code, several studies report that the same data analyzed with different software can lead to different results. I am seeing this problem as a manifestation of *deep software variability*: many factors (operating system, third-party libraries, versions, workloads, compile-time options and flags, *etc.*) themselves subject to variability can alter the results, up to the point it can dramatically change the conclusions of some scientific studies. In this keynote, I argue that deep software variability is a threat and also an opportunity for reproducible science. I first outline some works about (deep) software variability, reporting on preliminary evidence of complex interactions between variability layers. I then link the ongoing works on variability modelling and deep software variability in the quest for reproducible science.

EXTENDED ABSTRACT

Biology, medicine, physics, astrophysics, chemistry: all these scientific domains need to process large amount of data with more and more complex computations. For instance, studies about climate modelling and change involve the design of mathematical model, the mining and analysis of data, the executions of large simulations, *etc.* [2, 5, 6]. Both computational tasks require different kinds of software, from a set of scripts to automate the deployment to a comprehensive system containing several features that help researchers exploring various hypotheses. It is not an overstatement to say that computational science depends on software and its engineering [1, 7, 12].

One of the main promise of software is that a result obtained by an experiment (*e.g.*, a simulation) can be achieved again with a high degree of agreement. The quest for reproducibility takes different forms and requires to make all data and code available in such a way that the computations can be executed again with identical results. For achieving reproducible science, there are several challenges ahead involving multi-disciplinary collaboration and socio-technical innovation with software at the center of the problem. Despite the availability of data and code, several studies report that the same data analyzed with different software can lead to different results. For instance, applications of different analysis pipelines, alterations in software versions, and even changes in operating system have both shown to cause variation in the results of neuroimaging studies [3]. In [11] results and experience suggest that earth system models are not replicable under changes in the high-performance computing environment. Similar observations have been made in the machine learning or in the software engineering community [4]. As a result, software can threaten the scientific knowledge and recommendations built on top of these computations and studies.

I am seeing this problem as a manifestation of *deep software variability* [9]: many factors (operating system, third-party libraries, versions, workloads, compile-time options and flags, *etc.*) themselves subject to variability can alter the results, up to the point it can dramatically change the conclusions of some scientific studies. The idea of deep variability draws attention to the still little recognized fact that variability crosscuts the full computing stack, from high-level application software all the way down to hardware, and also along its evolution across time. In this keynote, I argue that deep software variability is a threat and also an opportunity for reproducible science.

I first outline some works about (deep) software variability. I report on preliminary evidence of complex interactions between variability layers. For instance, run-time options (*e.g.*, command line parameters) interact with compile-time options (*e.g.*, using `./configure`) with different effects of non-functional properties of a software [10]. There are various consequences w.r.t. tuning, default configurations, understanding, and testing of software. The best run-time configuration may not be optimal depending on the way the software has been compiled; the configuration knowledge about the run-time options depends on the compile-time options; a performance prediction model may not generalize and be pointless due to a different compilation. In [8], we conduct a large study over 8 configurable systems that quantifies the existing interactions between input data and configurations of software systems. The results exhibit that (1) inputs fed to software systems interact with their configuration options in non monotonous ways, significantly impacting their performance properties; (2) tuning a software system for its input data makes it possible to multiply its performance by up to ten (3) input variability can jeopardize the relevance of performance predictive models for a field deployment. There are also cases for which both compile-time, runtime, and input layer interact together, with notable changes in the performance of a software system [10].

I then link ongoing works about deep software variability in the quest of reproducible science. I revisit existing scientific studies and discuss the possible impacts of deep software variability. Software variability can be accidental: a change in some software parameters may incidentally change the conclusions. Software variability can also be an opportunity to generate and explore new hypotheses of a study. In-between, there is the striking question of whether results generalize and are robust to variability changes.

In both cases, the variability modelling community has a key role to play. Software is used increasingly to run scientific experiments and effective variability management can greatly improve the efficiency of experimental procedures, from problem definition, through experiment execution, to the robustness of results. We also need more inquiries and insights about deep software variability and their possible effects on computational scientific studies.

Acknowledgments. This research was funded by the ANR-17-CE25-0010-01 VaryVary project. I thank Luc Lesoil and Jean-Marc Jézéquel for feedbacks on early draft.

REFERENCES

[1] 2021. Hail, software! *Nature Computational Science* 1, 2 (01 Feb 2021), 89–89. https://doi.org/10.1038/s43588-021-00037-8

[2] Steve M Easterbrook and Timothy C Johns. 2009. Engineering the software for understanding climate change. *Computing in science & engineering* 11, 6 (2009), 65–74.

[3] Tristan Glatard, Lindsay B Lewis, Rafael Ferreira da Silva, Reza Adalat, Natacha Beck, Claude Lepage, Pierre Rioux, Marc-Etienne Rousseau, Tarek Sherif, Ewa Deelman, et al. 2015. Reproducibility of neuroimaging analyses across operating systems. *Frontiers in neuroinformatics* 9 (2015), 12.

[4] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[5] https://www.wcrp climate.org/. [n.d.]. World climat research program (WCRP).

[6] Christopher G Knight, Sylvia HE Knight, Neil Massey, Tolu Aina, Carl Christensen, Dave J Frame, Jamie A Kettleborough, Andrew Martin, Stephen Pascoe, Ben Sanderson, et al. 2007. Association of parameter, software, and hardware variation with large-scale behavior across 57,000 climate models. *Proceedings of the National Academy of Sciences* 104, 30 (2007), 12259–12264.

[7] Dorian Leroy, June Sallou, Johann Bourcier, and Benoit Combemale. 2021. When Scientific Software Meets Software Engineering. *Computer* (2021), 1–11. https://hal.inria.fr/hal-03318348

[8] Luc Lesoil, Mathieu Acher, Arnaud Blouin, and Jean-Marc Jézéquel. 2021. The Interaction between Inputs and Configurations fed to Software Systems: an Empirical Study. *CoRR* abs/2112.07279 (2021). arXiv:2112.07279 https://arxiv.org/abs/2112.07279

[9] Luc Lesoil, Mathieu Acher, Arnaud Blouin, and Jean-Marc Jézéquel. 2021. Deep Software Variability: Towards Handling Cross-Layer Configuration. In *VaMoS 2021 - 15th International Working Conference on Variability Modelling of Software-Intensive Systems*. Krems / Virtual, Austria, 1–8. https://hal.inria.fr/hal-03084276

[10] Luc Lesoil, Mathieu Acher, Xhevahire Tërnava, Arnaud Blouin, and Jean-Marc Jézéquel. 2021. The Interplay of Compile-time and Run-time Options for Performance Prediction. In *SPLC 2021 - 25th ACM International Systems and Software Product Line Conference - Volume A*. ACM, Leicester, United Kingdom, 1–12. https://doi.org/10.1145/3461001.3471149

[11] François Massonnet, Martin Ménégoz, Mario Acosta, Xavier Yepes-Arbós, Eleftheria Exarchou, and Francisco J Doblas-Reyes. 2020. Replicability of the EC-Earth3 Earth system model under a change in computing environment. *Geoscientific Model Development* 13, 3 (2020), 1165–1178.

[12] Greg Wilson, Dhavide A Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, et al. 2014. Best practices for scientific computing. *PLoS biology* 12, 1 (2014), e1001745.