Mind the Gap: General-Purpose Programming Languages Impede Scientific Model Development and Communication

Dominic Orchard
University of Cambridge & University of Kent

Climate science, and earth sciences more broadly, have a strong computational focus, with the development of executable models at the heart of much of their work. In many cases, these models become the primary description of the science itself, displacing more traditional mathematical formulations. Consequently, programming languages are now a core part of the computational scientist's toolbox. Within climate and earth sciences, Fortran remains the lingua franca for large-scale Earth System Models, while C++ is used occasionally and Python is becoming increasingly popular. More recently, Julia has become prominent through efforts such as the CliMA project, whose aim is to build a new open ESM platform using more modern techniques [2].

Increasingly there are calls for more open and FAIR (Findable, Accessible, Interoperable, Reusable) scientific software and data [12]. Scientists are increasingly sharing code, increasing the possibility of reusing, extending, and composing existing models. Relatedly, several communities have concluded that engineering effort must emphasize modularity and interoperability, e.g., in the Earth System Modeling Framework (ESMF) [3], the Modular Earth Submodel System (MESSy) [5], and the Basic Model Interface (BMI) and broader software ecosystem around CSDMS [11].

Our position, however, is that the general-purpose languages used for scientific modeling often impede the science because they do not readily enable scientists to express and exploit the domain-specific semantics of their models. A key example is dimensionality: physical equations are meaningful only when they are dimensionally consistent, yet the most widely used mainstream languages do not provide a standard way to express or statically enforce that consistency.[1] More generally, it is often difficult to attach additional scientific meaning to variables, such as their 'kind of quantity' or geographical scope (e.g., sea-surface temperature vs air temperature).

This missing expressivity shows up in naming conventions. For example, the Climate and Forecast (CF) conventions provide standardised names that capture domain-specific contexts for variables and datasets [1]. However, such names quickly become long and monolithic as they accumulate concepts; symbolic, atomic names do not compose! For example, there are 198 names associated with temperature (in v1.13 [1]). Four of these relate to air temperature:

- `air_temperature`
- `air_temperature_anomaly`
- `air_temperature_at_cloud_top`
- `air_temperature_at_effective_cloud_top_defined_by_infrared_radiation`

The result is that important scientific distinctions are forced into external naming schemes rather than being expressed (and checked) compositionally inside the programming language itself.

We therefore argue that new language designs are needed that are better suited to capturing the domain-specific meaning inherent in scientific models. This could include, but is not limited to, dimensionality, units-of-measure, kind-of-quantity, geographical location, and physical process assignment. One potential direction is to leverage intersection types [4] or extensible graded type systems so that properties can be layered compositionally, building on the insight that units-of-measure can be understood as graded monoids [7]. More broadly, this fits a longer computational-science agenda for programming-language research: if programs now serve as scientific artifacts, then languages should help expose scientific intent rather than obscure it [9]. This objective is also closely aligned with the original aspiration of object-oriented programming, that programming languages should be problem-oriented rather than computer-oriented [8], but with a focus on scientific problems rather than general software engineering problems.

[1] F# is a notable exception through its units-of-measure system [6]. Other dynamic approaches are possible and in C++ template-based approaches are maturing [10].

References

[1] CF Conventions and Metadata. Cf metadata conventions, version 1.13. https://cfconventions.org/Data/cf-conventions/cf-conventions-1.13/cf-conventions.html. Accessed: 2026-04-24.

[2] Climate Modeling Alliance. Clima: Climate modeling alliance. https://clima.caltech.edu/. Accessed: 2026-04-24.

[3] Chris Hill, Cecelia DeLuca, Max Suarez, ARLINDO Da Silva, et al. The architecture of the earth system modeling framework. *Computing in Science & Engineering*, 6(1):18–28, 2004.

[4] J Roger Hindley. Types with intersection: An introduction. *Formal Aspects of Computing*, 4(5):470–486, 1992.

[5] P. Jöckel, R. Sander, A. Kerkweg, H. Tost, and J. Lelieveld. Technical note: The modular earth submodel system (MESSy) - a new approach towards earth system modeling. *Atmospheric Chemistry and Physics*, 5(2):433–444, February 2005.

[6] Andrew Kennedy. Types for units-of-measure in f#: invited talk. In *Proceedings of the 2008 ACM SIGPLAN Workshop on ML*, ICFP08, pages 1–2. ACM, September 2008.

[7] Conor McBride and Fredrik Nordvall-Forsberg. *Type systems for programs respecting dimensions*, pages 331–345. World Scientific, January 2022.

[8] Kristen Nygaard and Ole-Johan Dahl. The development of the SIMULA languages. *History of Programming Languages*, pages 439–480, June 1978.

[9] Dominic A. Orchard and Andrew C. Rice. A computational science agenda for programming language research. In David Abramson, Michael Lees, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Proceedings of the International Conference on Computational Science, ICCS 2014, Cairns, Queensland, Australia, 10–12 June, 2014*, Procedia Computer Science, pages 713–727. Elsevier, 2014.

[10] Mateusz Pusz. mp-units. https://mpusz.github.io/mp-units/latest/. Documentation site. Accessed: 2026-04-24.

[11] Gregory E. Tucker, Eric W. H. Hutton, Mark D. Piper, Benjamin Campforts, Tian Gan, Katherine R. Barnhart, Albert J. Kettner, Irina Overeem, Scott D. Peckham, Lynn McCready, and Jaia Syvitski. CSDMS: a community platform for numerical modeling of earth surface processes. *Geoscientific Model Development*, 15(4):1413–1439, February 2022.

[12] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. ́t Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1), March 2016.