https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Code accessibility and code quality across phases of the models of the Coupled Model Intercomparison Project

Michael García-Rodríguez[1,2], Javier Rodeiro-Iglesias[2], and Juan A. Añel[1]

[1]EPhysLab, CIM-UVigo, Universidade de Vigo, 32004, Ourense, Spain
[2]School of Computer Sciences, Universidade de Vigo, 32004, Ourense, Galicia, Spain

Correspondence: Michael García-Rodríguez (micgarcia@uvigo.gal) and Juan A. Añel (j.anhel@uvigo.gal)

Abstract. This study extends previous research on CMIP5 models to investigate the reproducibility of climate models within the Coupled Model Intercomparison Project (CMIP). It evaluates the accessibility to the source code of the CMIP models through all their phases, emphasizing the need for public repositories to ensure transparency regarding model input, output, and usage rights, along with an analysis of licenses for compliance with scientific standards. A central focus of the research is the assessment of code quality against best practices. In addition, the study examines the historical evolution of computational and code quality across various phases of CMIP, highlighting progress and improving traceability to support scientific reproducibility. We provide valuable insights for future research, proposing solutions and tools designed to improve replicability and enhance project lifecycles that are applicable not only to CMIP but also to broader scientific contexts.

1 Introduction

Reproducibility is a cornerstone of science and the scientific method, and the challenge of achieving reproducible science when computational methods are involved is well known (Añel, 2011; Costa et al., 2024; Wu et al., 2024). In recent years, it has been a topic that is recurrently discussed by the scientific community (e.g., Allison et al. (2018); Stodden et al. (2018)).

Ensuring computational scientific reproducibility (CSR) requires addressing a variety of issues when designing and using models, particularly in climate models (Añel, 2017). Among these issues are legal aspects of software distribution and intellectual property, which are often unknown to researchers. Several studies have revealed very low levels of CSR (Allison et al., 2018; Stodden et al., 2018). Efforts to address the problem have included informal initiatives, such as documenting the accessibility of climate models (Easterbrook, 2009; RealClimate.org, 2009), as well as more formal assessments that evaluate model quality and transparency (e.g., Pipitone and Easterbrook (2012); Añel et al. (2021)).

Improving the CSR is possible through a range of actions. For example, journals may adopt code and data policies (Stodden et al., 2013; Geosc. Model Dev. Editors, 2015; Nature, 2018), while resarchers can follow recommended practices such as providing complete software documentation, structuring code into functions or modules, and publishing code and data in trusted repositories that ensure long-term preservation with permanent identifiers such as a Digital Object Identifier (DOI) (Wilson et al., 2017).

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

In some research fields, it is common practice to publish detailed methodological information. However, this habit is far less widespread in the development, maintenance, and scientific reproducibility of climate models (Añel et al., 2021). It remains concerning that the underlying model code is often not available, especially given its fundamental role in scientific research. Beyond basic scientific transparency, there are additional reasons to support the availability of climate models code, including the need to preserve knowledge about the development cycles of these models. Moreover, previous studies have shown that climate models frequently lack adequate documentation (Wieters and Fritzsch, 2018).

Climate models have evolved over decades, from early attempts that simulated basic processes (radiative balance, convection, terrestrial rotation, etc.) to comprehensive systems that represent interactions among major subsystems such as the atmosphere, ocean, and land. In many cases, these early models gradually developed into the sophisticated models used today. Originally, research and development of models was undertaken independently by research groups around the world. However, in 1995, a collaborative approach emerged that aimed to share progress and results among groups, thus contributing to a better understanding of the problem of global climate modelling and climate change. This initiative became the 'Coupled Model Intercomparison Project' (CMIP, 2025a) (CMIP), which is now in its seventh phase (CMIP7). Over the years, CMIP has undergone several phases, and its evolution (CMIP, 2025b) has raised several methodological questions, some of which had previously been discussed in other research fields, and adapted here to climate modelling. These include to define standardized experimental protocols, how to ensure reproducibility of simulations, or how to manage uncertainty and ensemble approaches.

In parallel with the development of climate models, software design practices have also advanced. Since the early days of software, code quality has been a major concern, and numerous approaches to asses it were proposed more than forty-five years ago (e.g. McCall et al. (1977); Boehm (1986)). While many of the early ideas about software quality are now outdated due to changes in computer architectures, software quality metrics have continued to evolve to address current needs.

More recently, the quality of software used in science and engineering has become a critical issue, given its direct impact on reproducibility and replicability (Association for Computing Machinery, 2020). Ensuring that scientific results can be reproduced requires not only high-quality code but also its availability and proper documentation. The lack of formal programming training among scientists, often leading to software-related challenges (Baxter et al., 2006), has long been recognized as a barrier to achieving this goal. Nevertheless, scientific software development has improved as researchers adopt best practices from broader software engineering fields (Nguyen-Hoan et al., 2010; Arvanitou et al., 2021).

Previous studies have shown that scientific software still has substantial room for improvement (Kanewala and Bieman, 2014a; Trisovic et al., 2022). Consequently, several authors have proposed best practices to enhance scientific software quality, development workflows, and code accessibility (Wilson et al., 2014; Riesch et al., 2020; Hunter-Zinck et al., 2021). However, the issue remains complex. Early software development faced strict hardware limitations, which required careful resource optimization. Over time, exponential increases in computing power have often masked the consequences of suboptimal code (Moore, 1965). With the widespread use of outsourced computing resources and cloud technologies in scientific research and climate modelling (Añel et al., 2020), where resources may appear virtually unlimited, the perceived impact of inefficient code is diminished. For instance, climate modelling experiments often run on supercomputers for months and consume vast computational resources, which may unintentionally reinforce this perception.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Given these challenges, this work provides an analysis of the current state of the art of the accessibility and static code quality of the CMIP models across all their historical phases, from CMIP1 to CMIP6. This study complements existing research in the field (Durack et al., 2025) and extends our previous efforts focused on models from the fifth phase of the CMIP (CMIP5) (Añel et al., 2021), a broader analysis repeatedly requested by some of our research peers in the climate modelling community. Section 2 describes the Methods used in this work, Section 3 evaluates the accessibility to the model code, Section 4 examines the programming language used in CMIP models, Section 5 presents results from static code analysis for the different models, and finally we offer a discussion and conclusions.

2 Methods

To verify the availability of the models, we employed a systematic approach similar to that described in Añel et al. (2021). Our first step consisted of checking access to the source code of the model, its configuration, and the associated documentation, including details on parameters, input data, and other relevant components. We primarily relied on the official download links provided on the CMIP website. When direct access was not available, we search for contact information through institutional websites or broader Internet searches. When necessary, we attempted to communicate the responsible institution or researchers via email and telephone. If access request for a given model‘s code was denied, we followed up with a survey to better understand the reasons for the refusal. Unfortunately, we did not receive any responses. For models developed during the earliest CMIP phases, obtaining access was often impossible, as many of these versions had not been adequately preserved.

As in Añel et al. (2021), this process revealed several barriers to transparency and reproducibility. For example, in some cases, the only available contact information was found in publications behind paywalls or embedded within metadata of NetCDF output files. Accessing such metadata requires technical skills and computational resources that are often beyond the reach of the general public and even some scientists.

For each climate model obtained, the licences associated with the model and potential restrictions to its use were verified. We also verified whether the model distribution included practical information on how to compile, configure, and execute the code to run an experiment. From a reproducibility perspective, this type of documentation is a very important aspect.

Furthermore, we analyse the evolution of the code quality across CMIP phases.To do so, we applied static code analysis, a method for evaluating programs without executing or compiling them. Specifically, we used the Fortran static code analysis tool FortranAnalyser v2.0 (http://fortrananalyser.ephyslab.uvigo.es/) (García-Rodríguez, 2022; García-Rodríguez et al., 2024). We selected this tool because Fortran is the dominant programming language in the CMIP models (as demonstrated later), and because alternative tools have limited applicability for assessing scientific code written in Fortran (Kanewala and Bieman, 2014b).

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

3 Availability and Reproducibility

Following the attempts made, successful access was obtained for 59 of the 262 individual models contributing to all CMIP phases (see Figure 1 and Table A1). These correspond to a total of 18 coupled models out of the 121 existing ones: one from CMIP3, ten from CMIP5, and seven from CMIP6. For CMIP1 and CMIP2, no model could be recovered, as their code had not been preserved over time. CMIP7 models were not included in this study because this is the current CMIP phase of the project, and active developement and usage are still ongoing.

A pronouced regional bias emerged in the geographic distribution of accesible models. The United States of America, France, Germany, and Norway stand out as the most open contributors; in all cases, institutions granted access to the full set of models they developed, independently of the CMIP phase. This, however, remained conditional on the availability of older model versions that institutions had effectively preserved. We hypothesize that national or regional legalisation related to software copyright, intellectual property, and code disclosure may partly explain these differences.

Regarding previous analyses of CMIP5 accessibility, despite the fact that software can be patented in the United States, meanwhile it is not possible in the the European Union (van Wendel de Joode et al., 2003), we were able to obtain the source code for 10 out of the 28 coupled models contributed by U.S.-based research centres. In contrast, only 8 out of 34 EU-based models were obtained.

The potential explanations for this have not changed since the CMIP5 study. The involvement of U.S. federal employees in the development of some models may facilitate code release, as federal employees are legally required to make their work publicly available without restrictions (U.S. Code, 1976). Similarly, the development of NorESM, which incorporates substantial components from the U.S.-developed model CESM1 (Knutti et al., 2013), may have benefited from inherited code licensing, helping explain Norway's comparatively high accessibility rates. Conversely, decisions not to release model code may stem from legacy licensing constraints or the absence of copyright transfer agreements, which can prevent redistribution of older codebases.

To better visualize the regional distribution of accessible models, and illustrate differences in national policies governing code sharing, Figure 1 displays the percentage of models obtained per country. For models developed by EC-Earth and ECMWF, the corresponding data are displayed at a central European location (Slovakia), as these consortia involve between 12 and 23 European countries depending on the CMIP phase.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

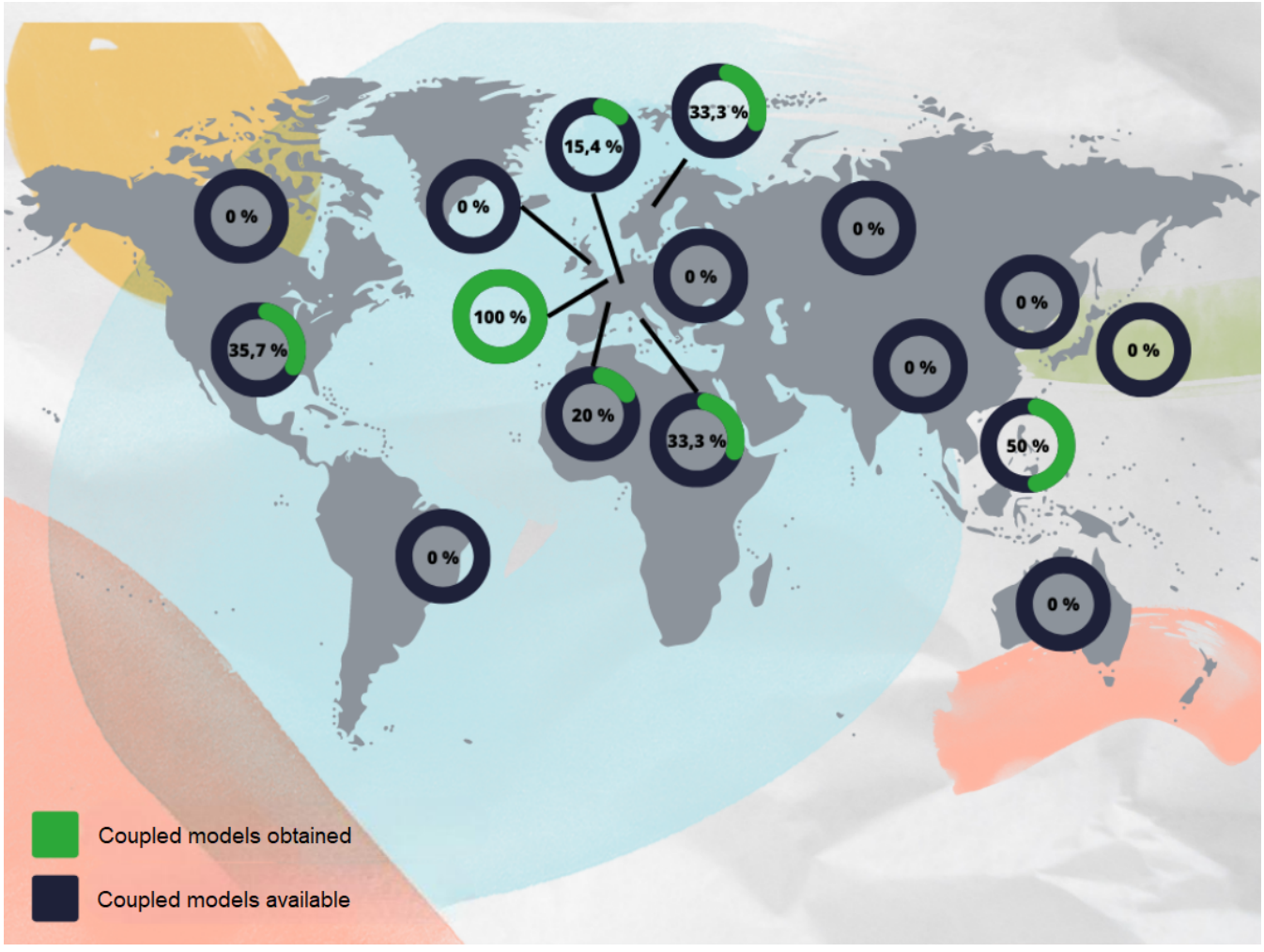

Figure 1. Geographical map with the percentage of coupled models obtained worldwide for all the CMIP phases existing so far and separated by the country in which they have been developed. The green colour and the percentages represent the coupled models obtained.

The current level accessibility to CMIP source codes remains critically limited. Only 59 out of the 121 coupled models were directly accessible; access to the remainder required active interaction, such as email correspondence with developers. As in previous work, even after identifying ourselves as researchers (addressing the potential concern that scientists might hesitate to share code with non-scientists), more than half of all model versions remain inaccessible.

There are several plausible reasons for limited code sharing among scientists and development teams. Some groups may face institutional or national legal restrictions preventing them from openly distributing source code. Others may be concerned about inheriting sharing rights for portions of the model code that were developed by different institutions or research groups, raising concerns about potential infringement on proprietary rights. In other cases, older models may no longer be available simply due to insufficient long-term maintenance of archives or repositories. In particular, when source code was obtained,

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

the developers did not provide explanations for their licensing choices; indeed, in some cases, the code lacked a clear license defining usage terms.

Although there is no evidence suggesting that scientific results derived from CMIP models are invalid, the lack of code availability can be misused to cast doubt on the reproducibility of the underlying scientific findings. For this reason, we strongly encourage modelling groups to adopt more robust practices that align with modern expectations of corporate social responsibility in science, particularly regarding reproducibility and replicability (CSR). Previous work has shown that sharing model code structures, even if currently suboptimal, can promote improvements in software quality and scientific transparency (Carlson et al., 2018).

Table 1: Coupled models whose code has been obtained, together with the phase to which they belong and the assigned reproducibility scores.The scoring mechanism was as follows: the maximum score of three stars was awarded to models that are accessible via the Internet without restrictions and with a licence that allows for full testing and evaluation of the model. The score was reduced by one star for each of the following criteria: if access to the model required contacting a researcher, centre or development group, signing licence agreements or identifying oneself as a scientist conducting research in climatology, as well as depending on the evaluation and use rights granted by the model licence (where applicable). An empty or unfilled star means that the model licence does not allow modification of the code.

Coupled Model	CMIP phase	Score
CCSM3	3	★★★
CFSv2	5	★★☆
IPSL	5	★★
MPI-ESM	5	★
GISS	5	★☆
GESO5	5	★☆
CCSM4	5	★★★
NorESM1	5	★★
NICAM	5	★
GFDL	5	★★
CESM1	5	★★★
HiRAM / TaiESM	6	★★★
CMCC	6	★★★
E3SM	6	★★★
MPI-ESM1.2	6	★

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Table 1 continuation of the previous page

Coupled Model	CMIP phase	Score
IPSL	6	★★
GISS	6	★☆
CESM	6	★★★

4 Programming languages used in CMIP Models

Over the course of different CMIP phases, progressive improvements in spatial resolution, numerical accuracy, and the inclusion of crucial climate processes have been incorporated into the contributing models (Drake, 2001). These advances have involved the continuous refinement of computational algorithms, the addition of new functionalities, and the systematic replacement of outdated components, all aimed at enhancing the scientific robustness and overall quality of model outputs.

In this context, understanding how models and their successive versions operate, envolve across CMIP phases, and adhere to established software development standards requires careful examination of the programming languages employed in their implementation. Programming langauges play a role in defining the structure, performance, and maintainability of climate models. Table 2 summarises the distribution of programming languages used across the models for which source code was available.

The choice of programming language in climate model development is strongly influenced by the scientific community behind each model, its computational requirements, and the degree of code inheritance involved. Climate models often emerge from long-term collaborative efforts involving multiple institutions and scientists disciplines, which can lead to the coexistence of multiple languages in a single project. While performance optimisation and scalability are essential considerations, given the high computational cost of climate simulations, such priorities are sometimes overridden by the need for rapid scientific deliverable.

To analyse the use of languages in the retrieved CMIP models, we employed the GitLab's repository analytics functionality (Zaporozhets, 2011), based on the GitHub 'linguist' library (Gardner, J. and GitHub staff, 2024). This functionality excludes non-relevant files and calculates the percentages of code used.

As shown in Table 2, Fortran (Backus, 1978; Metcalf et al., 2011) remains the dominant programming language in CMIP models. Its prevalence is well documented and reflects its efficiency in handling computationally intensive scientific calculations, as well as its long-standing stability in large legacy projects. The substantial multi-institutional investment in these models, often comprising millions of lines of code across multiple components (Pressman, 2005; McConnell, 2004; Fenton and Bieman, 2014) has made transitions to alternative languages both difficult and improbable. Consequently, Fortan has remained the primary choice to ensure consistency, compatibility, and continuity across successive model generations, with developers extending codebases rather than rewriting them.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Table 2. Proportion of programming languages used in all the climate models participating in the different phases of the CMIP. The analysis includes how much represents the documentation from the total of files provided with the codebase of the model.

	Fortran	Tex	C	Tcsh	Script	Makefile	R	HTML	Perl	CLisp	Shell	C++	PHP	Python	Documentation
CCSM3	81.6%	4.1%	3.5%	3.2%											7.6%
CFSv2-2011	9.4%		88.1%		2.4%	0.1%									
IPSL-CM5A	87.4%		1.9%			1.5%					7.7%				1.5%
MPI-ESM	83.5%	0.8%	11.5%		2.8%										1.4%
GISS-E2	89.8%				0.9%		4.1%	2.2%							3.0%
GEOS-5	75.6%			4.3%		4.4%			7.9%						7.8%
CCSM4	85.5%	6.1%		2.1%					3.3%						3.0%
NorESM	84.8%	5.3%						3.5%	2.3%						4.1%
NICAM.09	22.4%					0.1%			0.1%	77.4%					0.1%
GFDL	85.4%		4.7%					5.2%				2.5%	0.5%		1.7%
CESM1	87.5%	2.8%						4.5%	2.6%						2.6%
HiRAM	84.5%	2.8%						4.3%	2.5%						5.9%
CMCC	85.4%	2.6%						3.9%	2.4%					1.7%	4.0%
E3SM	90.9%	2.1%										3.7%		1.7%	1.6%
MPI-ESM1.2	52.7%		18.3%								13.9%	6.3%			8.8%
IPSL-CM6A	79.4%				4.4%				5.0%			4.0%			7.2%
GISS-E22	89.0%				1.0%		4.3%	2.5%							3.2%
CESM2	90.2%		0.4%						2.5%		0.4%			5.1%	1.4%

Python (Matthes, 2019) has increasing prominence in the climate modelling community, particularly for tasks such as as pre and post processing of model output, workflow scripting, rapid prototyping, and visualisation. Scripting languages such as as Perl and Bash are also used for workflow automation, job orchestration on HPC systems, and data management tasks. Additionally, R is frequently employed for statistical analysis and evaluation of observational and model generated datasets.

C and C++ (Kernighan and Ritchie, 1988; Stroustrup, 2013), are commonly used to optimise numerically demanding components or integrate specific modules requiring high performance. They often used alongside Fortran, either to improve the computational performance of specific subroutines or for compatibility with the use of other codes on which they depend. Notably, one of the models, CFSv2-2011, stands out as the only climate model in which C constitues the primary implementation language, accounting for 88.1% of its codebase.

An additional noteworthy case is that NICAM.09 has been developed predominantly in CLisp (Seibel, 2005). Nevertheless, Fortran remains the second most used language within NICAM.09, supporting its core dynamics and physics routines.

Based on these results and the overwhelming dominance of Fortran across the CMIP model suite, the present study focuses specifically on analysing the Fortran components of the obtained codebases.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

5 Code analysis

Here, we focus exclusively on the analysis of Fortran code, reflecting its longstanding and central role in climate model development. This choice is supported by the results in Table 2, which shows how Fortran constitutes a substantial portion of the programming languages used across the CMIP models.

To evaluate code quality, we employed FortranAnalyser, a static analysis tool designed to assess software written in any Fortran version. Its compatibility with legacy code is particularly important given the extensive historical layers present in climate models. Moreover, FortranAnalyser was developed ad-hoc for this purposes, addressing the absence of alternative tools capable of performing consistent, large scale static analysis on Fortran scientific software. Table 3 shows the static code quality scores produced by FortranAnalyser for each CMIP model analysed. While several CMIP6 models achieve comparatively high scores, the overall results do not reveal a clear or uniform trend of improvement across CMIP phases.

Table 3. Scores, over possible maximum score of ten points, of static code quality for the CMIP models. For the CMIP6, CESM1 and CESM2 are evuated as a single codebase.

Phase	Modeling Center	Model	Score
3	NCAR	CCSM3	4.014
5	COLA & NCEP	CFSv2-2011	2.920
5	IPSL	IPSL-CM5	3.974
5	MPI-M	MPI-ESM	3.863
5	NASA-GISS	GISS-E2	2.774
5	NASA-GMAO	GEOS-5	4.131
5	NCAR	CCSM4	3.536
5	NCC	NorESM1-M	3.703
5	NICAM	NICAM.09	4.341
5	NOAA-GFDL	GFDL	3.179
5	NSF-DOE-NCAR	CESM1	3.929
6	AS-RCEC	HiRAM	4.317
6	CMCC	CMCC-CM2	4.274
6	E3SM-Project	E3SM	4.274
6	HAMMOZ-Consortium	MPI-ESM1.2-HAM	2.790
6	IPSL	IPSL-CM6	4.643
6	NASA-GISS	GISS-E2	2.826
6	NCAR	CESM1 CESM2	3.970

While some models in CMIP6 exhibit high static code quality scores, the results do not clearly demonstrate a consistent trend of improvement across CMIP phases.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

In CMIP Phase 3, only one model, CCSM3, was available, receiving a score of 4.014. Although this score is relatively high, the absence of additional models from the same phase lacks contextual interpretation. Moving to Phase 5, the number of evaluated models increases significantly, yielding a wider spread of the code quality. Models such as GISS-E2 (2.774) and CFSv2-2011 (2.920) score relatively low, indicating potential structural or maintainability weaknesses in their codebases. In contrast, NICAM.09 (4.341) and GEOS-5 (4.131) show stronger performance, highlighting heterogeneity in coding practices across modelling groups during this phase.

A particularly notable case is the IPSL model. In Phase 5, IPSL-CM5 achieved a solid score of 3.974, but its successor on phase 6, IPSL-CM6, reached a score of 4.643, representing a substantial qualitative leap. This progression demonstrates how targeted software quality practices, coupled with the application of static analysis tools, can lead to measurable gains in the maintainability and robustness of scientific models. A deeper evaluation of the development practices and analysis results for this model is available in García-Rodríguez et al. (2024), providing insight into how these tools contribute to enhanced code transparency and structure.

In general, Phase 6 models show higher overall scores, indicating an upward trajectory in software quality. Models such as HiRAM (4.317), CMCC-CM2 (4.230), and E3SM (4.118) reflect a more mature approach to model development, likely supported by better development workflows, more formalized coding standards, and increased awareness of reproducibility and sustainability requirements.

Taken together, the observed improvements across CMIP phases suggests that software quality is increasingly being recognized as an integral component of scientific modelling. Static analysis tools not only provides objective metrics for evaluating code quality but also deliver actionable insights for improving code programming bases. As demonstrated by the IPSL case, integrating that tools into model development pipelines can yield tangible benefits and should be more widely encouraged throughout the climate modelling community.

6 Discussion

Future climate modelling efforts should incorporate the findings presented in this study. Researchers developing new models from scratch, and therefore not constrained by legacy code, are particularly well positioned to implement appropriate licensing practices, comprehensive documentation, and reproducibility protocols from the outset.

It is advisable that final, production-ready versions of climate models used to support conclusions in international climate assessments, together with their corresponding simulation outputs, be archived in long-term, trusted repositories. This recommendation applies broadly to all model intercomparison initiatives. Given the central role of climate models in assessing climate change, restricted access to source code or configuration details may be perceived as a methodological vulnerability. The inability to reproduce published results due solely to missing code or insufficient documentation represents a significant barrier to scientific transparency. It is also crucial to emphasize the distinction between reproducibility and replicability (Association for Computing Machinery, 2020): reproducibility refers to obtaining identical results under identical conditions,

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

whereas replicability refers to achieving consistent results under varied conditions. Both dimensions are essential for scientific reliability and trust.

The combination of final model releases with modern computational solutions, including cloud infrastructures, containerization technologies, and workflow automation, provides a promising avenue toward achieving full replicability in Earth system modelling (Perkel, 2019; Añel et al., 2020). Such approaches have already demonstrated their value in other scientific domains by reducing technical barriers and fostering more robust computational environments.

At the same time, climate model evaluation frameworks continue to evolve. Validation tools employing a wide range of performance metrics are increasingly used to assess model output. In this context, tools such as ESMValTool (Eyring et al., 2020), which was specifically designed for systematic evaluation, offer an important opportunity: to integrate code accessibility, licensing, and code quality evaluation assessments into the CMIP evaluation workflow. Incorporating these dimensions would encourage more consistent standards across modelling centres and improve transparency.

Scientific gaps identified in CMIP5 (Stouffer et al., 2017) further highlight the need for open-source policies. Without access to the underlying source code, it becomes impossible to diagnose discrepancies between models or resolve differences in methodological implementation, ultimately hindering scientific progress. Experience in other areas of software development shows that open code sharing contributes to better implementation practices, improved reliability, and more efficient error detection (Boulanger, 2005; DoD CIO, 2009). Moreover, open collaboration supports the collective scientific effort required to address climate change (Easterbrook, 2010) and aligns with core principles of scientific integrity (Añel, 2019).

Funding agencies and research institutions therefore have a critical role to play. Allocating resources not only for model development but also for documentation, reproducibility workflows, and long-term code preservation is essential for ensuring that climate modelling keeps pace with evolving expectations around transparency and open science.

Although challenges related to computational scientific reproducibility (CSR) persist across scientific disciplines, these challenges also present significant opportunities. In a field as societally vital as climate research, addressing the structural and organizational barriers identified here will strengthen the robustness, credibility, and long-term sustainability of the climate modelling enterprise.

7 Conclusions

This study provides a systematic evaluation of source code accessibility and static code quality across all historical phases of the Coupled Model Intercomparison Project (CMIP). Our analysis confirms that Fortran remains the predominant programming language in climate model development, reflecting its long-standing role in scientific computing and its extensive legacy within Earth system modelling frameworks.

A central outcome of this work is the persistent lack of accessibility to the source code of most CMIP models. This limitation continues to hamper reproducibility, a principle explicitly emphasised in current scientific standards and journal policies. The restricted availability of model code arises from several factors, including licensing constraints, dependencies on proprietary components, and the incomplete preservation of early CMIP model versions. Additionally, structural aspects of the research

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

environment—such as production-oriented publication pressures and the low prioritisation of long-term repository maintenance—further exacerbate these issues. Expanding the use of robust, long-term digital preservation platforms and integrating validation tools throughout the development cycle would meaningfully strengthen the transparency of climate modelling.

To enhance reproducibility and replicability, it is essential that final, fully documented versions of climate models be published in stable public repositories. We also recommend incorporating code analysis tools such as FortranAnalyser into CMIP evaluation workflows to improve the traceability, maintainability, and robustness of model implementations. Furthermore, adopting FLOSS (free-libre open-source software) practices and allocating dedicated funding to support collaborative development would facilitate broader community engagement and harmonisation of software standards.

Based on the findings of this work, several priority directions emerge for the CMIP community:

- **Improving code accessibility:** Overcoming licensing barriers, minimising reliance on proprietary dependencies, and promoting deposition of code and documentation in long-term repositories (e.g. Zenodo) are necessary steps. Establishing clear requirements for model availability within international assessments would also support reproducible experimental design.
- **Adopting FLOSS practices:** The publication of model code under permissive licences, together with comprehensive documentation and the use of collaborative development platforms, would enhance transparency and facilitate cross-institutional contributions.
- **Advancing code analysis tools:** Continued development of tools such as FortranAnalyser is essential to identify structural issues, support consistent programming practices, and provide actionable metrics for quality improvement across modelling groups.
- **Integrating validation tools into evaluation workflows:** Systematic use of validation frameworks can improve software reliability by enabling early detection of regressions and facilitating reproducible assessment of model updates.
- **Strengthening transparency and reproducibility:** Establishing clear standards for publishing code, data, and configuration details is key to ensuring that climate model results can be independently verified and meaningfully compared.

The challenges identified in this study are technical, organisational, and cultural. Addressing them requires coordinated action across modelling centres, infrastructure providers, and the broader scientific community. Improving code accessibility, software quality, and transparency will enhance the reliability of climate models and reinforce their role as essential tools for understanding and projecting climate change.

Code and data availability

The code of the models analysed in this work and the reports on code quality generated for each one of the models is deposited in the following URLs except for CCSM3, CCSM4, CESM1, MPI-ESM, NICAM.09, NorESM1-M, CMCC-CM2, E3SM, and MPI-ESM1.2-HAM for which we do not have the right to redistribute them:

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

- cfsv2 cmip5: https://zenodo.org/records/16932623
- geso5 cmip5: https://zenodo.org/records/16932661
- gfdl cmip5: https://zenodo.org/records/16932683
- giss cmip5: https://zenodo.org/records/16932705
- ipsl cmip5: https://zenodo.org/records/16932753
- cesm1 cmip6: https://zenodo.org/records/16933339
- giss cmip6: https://zenodo.org/records/16933370
- hiram cmip6: https://zenodo.org/records/16932804
- ipsl cmip6: https://zenodo.org/records/16933458

The FortranAnalyser software used here is deposited in https://doi.org/10.5281/zenodo.5942943 (García-Rodríguez (2022)).

8 Competing interests

At least one of the (co-)authors is a member of the editorial board of Geoscientific Model Development.

9 Author contribution

MGR carried out the formal analysis, investigation, data curation, and software development. JAA, JRI, and M.G.R. contributed to the conceptualization and validation of the study. JAA and JRI were responsible for funding acquisition, project administration, resources, and supervision. JAA and JRI also contributed to the formulation and evolution of overarching research goals and aims. MGR prepared the original draft of the manuscript, and all authors contributed to writing, review, and editing.

10 Acknowledgements

J.A.A. was supported through two grants from the Spanish Research Agency (CHESS, PID2021-124991OB-I00 and SARTRE, PID2024-158326NB-I00). The EPhysLab is supported by the Government of Galicia (Grant: GRC-ED431C 2025/37).

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Table A1: List of the 262 models that are part of the Coupled Model Intercomparison Project, indicating the phase each one corresponds to, modeling center, and model name, highlighting in green those for which the model code has been obtained.

CMIP Phase	Modeling center	Country	Model	Downloaded
1	Bureau of Meteorology Research Centre	Australia	BMRC1	No
1	CCCma	Canada	CCCma	No
1	Centre for Climate Systems Research	Japan	CCSR	No
1	CERFACS	France	CERFACS1	No
1	COLA	USA	COLA 1	No
			COLA 2	No
1	CSIRO	Australia	CSIRO	No
1	MPI	Germany	ECHAM1+LSG	No
			ECHAM3+LSG	No
			ECHAM4+OPYC3	No
			ECHAM4+HOPE-G	No
1	GFDL	USA	GFDL_R15_a	No
1	GISS	USA	GISS(Miller)	No
			GISS(Russell)	No
1	LASG	China	IAP/LASG1	No
1	IPSL	France	LMD/IPSL1	No
1	MRI	Japan	MRI1	No
1	NCAR	USA	NCAR(CSM)	No
			NRL1	No
1	UKMO	United Kingdom	UKMO (HadCM2)	No
2	CERFACS	France	ARPEGE/OPA2	No
2	BMRC	Australia	BMRCa	No
2	CCSR/NIES	Japan	CCSR/NIES	No
			CCSR/NIES2	No

Table A1 continued from previous page

2	CCCma	Canada	CGCM1	No
			CGCM2	No
2	CSIRO	Australia	CSIRO Mk2	No
2	NCAR	USA	CSM 1.0	No
			CSM 1.3a	No
2	DKRZ	Germany	ECHAM3/LSG	No
2	MPI	Germany	ECHAM4/OPYC	No
2	GFDL	USA	GFDL_R15_a	No
			GFDL_R15_b	No
			GFDL_R30_c	No
2	GISS	USA	GISS2	No
2	IAP/LASG	China	GOALS	No
2	UKMO	United Kingdom	HadCM2	No
			HadCM3	No
2	IPSL/LMD	France	IPSL-CM2	No
2	MRI	Japan	MRI1	No
			MRI2	No
2	NCAR	USA	DOE PCM	No
3	BCC	China	BCC-CM1	No
3	BCCR	Norway	BCCR-BCM2.0	No
3	NCAR	USA	CCSM3	Yes
3	CCMA	Canada	CGCM3.1(T47)	No
			CGCM3.1(T63)	No
3	Météo France & CNRM	France	CNRM-CM3	No
3	CSIRO	Australia	CSIRO-Mk3.0	No
			CSIRO-Mk3.5	No
3	MPI	Germany	ECHAM5/MPI-OM	No
3	MI of the University of Bonn , MRI of KMA	Germany	ECHO-G	No

Table A1 continued from previous page

3	LASG / Institute of Atmospheric Physics	China	FGOALS-g1.0	No
3	NOAA	USA	GFDL-CM2.0	No
			GFDL-CM2.1	No
3	NASA	USA	GISS-AOM	No
			GISS-EH	No
			GISS-ER	No
3	Instituto Nazionale di Geofisica e Vulcanologia	Italy	INGV-SXG	No
3	Institute for Numerical Mathematics	Russia	INM-CM3.0	No
3	IPSL	France	IPSL-CM4	No
3	CCSR, NIES and FRCGC	Japan	MIROC3.2(hires)	No
			MIROC3.2(medres)	No
3	MRI	Japan	MRI-CGCM2.3.2	No
3	NCAR	USA	PCM	No
3	HCCPR and Met Office	United Kingdom	UKMO-HadCM3	No
			UKMO-HadGEM1	No
5	BCC	China	BCC-CSM1.1	No
			BCC-CSM1.1(m)	No
5	CCCma	Canada	canAM4	No
			canCM4	No
			canESM2	No
5	CMCC	Italy	CMCC-CESM	No
			CMCC-CM	No
			CMCC-CMS	No
5	CNRM-CERFACS	France	CNRM-CM5	No
			CNRM-CM5-2	No
5	COLA and NCEP	USA	CFSv2-2011	Yes

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Table A1 continued from previous page

5	CSIRO-BOM	Australia	ACCESS1.0	No
			ACCESS1.3	No
5	CSIRO-QCCCE	Australia	CSIRO-Mk3.6.0	No
5	EC-EARTH	Sweden, the Netherlands, Denmark, Spain, Ireland, Italy Finland, Germany, Portugal, Greece, Norway and Belgium	EC-EARTH	No
5	FIO	China	FIO-ESM	No
5	GCESS	China	BNU-ESM	No
5	INM	Russia	INM-CM4	No
5	IPSL	France	IPSL-CM5A-LR	Yes
			IPSL-CM5A-MR	Yes
			IPSL-CM5B-LR	Yes
5	LASG-CESS	China	FGOALS-g2	No
5	LASG-IAP	China	FGOALS-g1	No
			FGOALS-s2	No
5	MIROC	Japan	MIROC4h	No
			MIROC5	No
			MIROC-ESM	No
			MIROC-ESM-CHEM	No
5	MOHC	United Kingdom	HadCM3	No
			HadCM3Q	No
			HadGEM2-A	No
			HadGEM2-CC	No
			HadGEM2-ES	No
			MPI-ESM-LR	Yes
			MPI-ESM-MR	Yes

Table A1 continued from previous page

5	MPI-M	Germany	MPI-ESM-P	Yes
5	MRI	Korea	MRI-AGCM3.2H	No
			MRI-AGCM3.2S	No
			MRI-ESM1	No
			MRI-CGCM3	No
5	NASA-GISS	USA	GISS-E2-H	Yes
			GISS-E2-H-CC	Yes
			GISS-E2-R	Yes
			GISS-E2-R-CC	Yes
5	NASA-GMAO	USA	GEOS-5	Yes
5	NCAR	USA	CCSM4	Yes
5	NCC	Norway	NorESM1-M	Yes
			NorESM1-ME	Yes
5	NICAM	Netherlands	NICAM.09	Yes
5	NIMR/KMA	United Kingdom	HadGEM2-AO	No
5	NOAA-GFDL	USA	GFDL-CM2.1	Yes
			GFDL-CM3	Yes
			GFDL-ESM2G	Yes
			GFDL-ESM2M	Yes
			GFDL-HIRAM-C180	Yes
			GFDL-HIRAM-C360	Yes
5	NSF-DOE-NCAR	USA	CESM1(BGC)	Yes
			CESM1(CAM5)	Yes
			CESM1(CAM5.1,FV2)	Yes
			CESM1(FASTCHEM)	Yes
			CESM1(WACCM)	Yes
6	AER	EEUU	LBLRTM 12.8	No
			RRTMG-LW 4.91	No
			RRTMG-SW 4.02	No

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Table A1 continued from previous page

6	AS-RCEC	Taiwan	HiRAM-SIT-HR	Yes
			HiRAM-SIT-LR	Yes
			TaiESM 1.0	Yes
6	AWI (MPI)	Germany	AWI-CM 1.1 HR	No
			AWI-CM 1.1 LR	No
			AWI-CM 1.1 MR	No
			AWI-ESM 1.1 LR	No
			AWI-ESM 2.1 LR	No
6	BCC	China	BCC-CSM 2 HR	No
			BCC-CSM 2 MR	No
			BCC-ESM 1	No
6	BNU	China	BNU-ESM 1.1	No
6	CAMS	China	CAMS-CSM 1.0	No
6	CAS	China	CAS-ESM 2.0	No
			FGOALS-f3-H	No
			FGOALS-f3-L	No
			FGOALS-g3	No
6	CCCma	Canada	CanESM5	No
			CanESM5-CanOE	No
6	CCCR-IITM	EEUU	IITM-ESM	No
6	CMCC	Italy	CMCC-CM2-HR4	No
			CMCC-CM2-SR5	No
			CMCC-CM2-VHR4	No
			CMCC-ESM2	No
6	CNRM-CERFACS	France	CNRM-CM6-1	No
			CNRM-CM6-1-HR	No
			CNRM-ESM2-1	No
			CNRM-ESM2-1-HR	No
6	CSIR-Wits-CSIRO	Australia	VRESM 1.0	No

Table A1 continued from previous page

6	CSIRO	Australia	ACCESS-ESM1.5	No
6	CSIRO-ARCCSS	Australia	ACCESS-CM2	No
6	E3SM-Project	EEUU	E3SM 1.0	Yes
			E3SM 1.1 ECA	Yes
			E3SM 1.1	Yes
6	EC-Earth-Consortium	Sweden, the Netherlands, Denmark, Spain, Ireland, Italy Finland, Germany, Portugal, Greece, Norway and Belgium	EC Earth 3.3	No
			EC-Earth3-AerChem	No
			EC-Earth3-CC	No
			EC-Earth3-GrIS	No
			EC-Earth3-HR	No
			EC-Earth3-LR	No
			EC-Earth3-Veg	No
			EC-Earth3-Veg-LR	No
			EC-Earth3P	No
			EC-Earth3P-HR	No
			EC-Earth3P-VHR	No
6	ECMWF	Austria, Belgium, Croatia, Denmark, Estonia, Finland, France, Germany,	ECMWF-IFS-HR	No
			ECMWF-IFS-LR	No
			ECMWF-IFS-M	No
6	FIO-QLNM	China	FIO-ESM 2.0	No
6	HAMMOZ-Consortium	Germany	MPI-ESM1.2-HAM	Yes
6	INM	Ireland, Italy, Luxembourg, Netherlands, Norway, Poland,	INM-CM4-8	No
			INM-CM5-0	No
			INM-CM5-H	No
6	INPE	Brazil	BESM 2.9	No
			4AOP-v1-5	Yes
			IPSL-CM5A2-INCA	Yes
			PSL-CM6A-ATM-HR	Yes
			IPSL-CM6A-LR	Yes

Table A1 continued from previous page

6	IPSL	Edafrom	IPSL-CM6A-LR-INCA	Yes
			IPSL-CM7A-ATM-HR	Yes
			IPSL-CM7A-ATM-LR	Yes
6	KIOST	Korea	KIOST-ESM	No
6	MESSy-Consortium	Germany	EMAC-2-53-Vol	No
			EMAC-2-54-AerChem	No
6	MIROC	Japan	MIROC-ES2H	No
			MIROC-ES2H-NB	No
			MIROC-ES2L	No
			MIROC6	No
			NICAM16-7S	No
			NICAM16-8S	No
			NICAM16-9D-L78	No
			NICAM16-9S	No
6	MOHC	United Kingdom	HadGEM3-GC31-LM	No
			HadGEM3-GC31-MH	No
			HadGEM3-GC31-MM	No
6	MOHC NERC	United Kingdom	HadGEM3-GC31-HH	No
			HadGEM3-GC31-HM	No
			HadGEM3-GC31-LL	No
			UKESM1.0-MMh	No
			UKESM1.ice-LL	No
6	MOHC NERC NIMS-KMA NIWA	United Kingdom	UKESM1.0-LL	No
6	MPI-M AWI	Germany	ICON-ESM-LR	No
			MPI-ESM1.2-XR	No
			MPI-ESM1.2-LR	No
6	MPI-M DWD DKRZ	Germany	MPI-ESM1.2-HR	No
			MRI-AGCM3-2-H	No

Table A1 continued from previous page

6	MRI	Japan	MRI-AGCM3-2-S	No
			MRI-ESM2.0	No
6	NASA-GISS	EEUU	GISS-E2.1G	Yes
			GISS-E2-1-G-CC	Yes
			GISS-E2.1H	Yes
			GISS-E2-2-G	Yes
			GISS-E3-G	Yes
6	NCAR	EEUU	CESM1-1-CAM5-CMIP5	Yes
			CESM1-CAM5-SE-HR	Yes
			CESM1-CAM5-SE-LR	Yes
			CESM2	Yes
			CESM2-FV2	Yes
			CESM2-SE	Yes
			CESM2-WACCM	Yes
			CESM2-WACCM-FV2	Yes
6	NCC	Norway	NorCPM1	No
			NorESM1-F	No
			NorESM2-HH	No
			NorESM2-LM	No
			NorESM2-LME	No
			NorESM2-LMEC	No
			NorESM2-MH	No
			NorESM2-MM	No
6	NIMS-KMA	Korea	KACE1.0-G	No
6	GFDL	EEUU	GFDL-AM4	No
			GFDL-CM4	No
			GFDL-CM4C192	No
			GFDL-ESM2M	No
			GFDL-ESM4	No

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Table A1 continued from previous page

6	NOAA-GFDL			GFDL-GLOBAL-LBL	No
				GFDL-GRTCODE	No
				GFDL-OM4p5B	No
				GFDL-RFM-DISORT	No
6	NTU	Taiwan		TaiESM1-TIMCOM	No
6	NUIST	China		NESM v3	No
6	PCMDI	Germany		PCMDI-test 1.0	No
6	PNNL-WACCEM	EEUU		CAM-MPAS-HR	No
6	RTE-RRTMGP-Consortium	EEUU		RTE+RRTMGP (2018-12-04 full-resolution)	No
6	SNU	Korea		SAM0-UNICON	No
6	THU	China		CIESM	No
6	UA	EEUU		MCM-UA-1-0	No
6	UCI	EEUU		CESM1-WACCM-SC	No
6	UHH	Germany		ARTS 2.3	No
6	UofT	Canada		UofT-CCSM4	No
6	UTAS	Australia		CSIRO Mk3L 1.3	No

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

References

Añel, J. A.: The Importance of Reviewing the Code, Communication of the ACM, 54, 40–41, https://doi.org/10.1145/1941487.1941502, 2011.

Añel, J. A.: Comment on "Most computational hydrology is not reproducible, so is it really science?" by Hutton et al., Water Resourources Research, 53, 2572–2574, https://doi.org/10.1002/2016WR020190, 2017.

Añel, J. A.: Reflections on the Scientific Method at the beginning of the twenty-first century, Contemporary Physics, 1, 60–62, https://doi.org/10.1080/00107514.2019.1579863, 2019.

Añel, J. A., Montes, D. P., and Rodeiro Iglesias, J.: Cloud and Serverless Computing for Scientists, Springer, ISBN 978-3-03-041783-3, https://doi.org/10.1007/978-3-030-41784-0, 2020.

Añel, J. A., García-Rodríguez, M., and Rodeiro, J.: Current status on the need for improved accessibility to climate models code, Geoscientific Model Development, 14, 923–934, https://doi.org/10.5194/gmd-14-923-2021, 2021.

Allison, D., Shiffrin, R., and Stodden, V.: Reproducibility of research: Issues and proposed remedies, Proceedings of the National Academy of Sciences, 115, 2561–2562, https://doi.org/10.1073/pnas.1802324115, 2018.

Arvanitou, E.-M., Ampatzoglou, A., Chatzigeorgiou, A., and Carver, J. C.: Software engineering practices for scientific software development: A systematic mapping study, Journal of Systems and Software, 172, 110 848, https://doi.org/10.1016/j.jss.2020.110848, 2021.

Asociation for Computing Machinery: Artifact Review and Badging, available at: https://www.acm.org/publications/policies/artifact-review-and-badging-current (último acceso: 31/05/2024), 2020.

Backus, J.: The history of Fortran I, II, and III, p. 25–74, Association for Computing Machinery, New York, NY, USA, ISBN 0127450408, https://doi.org/10.1145/800025.1198345, 1978.

Baxter, S. M., Day, S. W., Fetrow, J. S., and Reisinger, S. J.: Scientific Software Development Is Not an Oxymoron, PLOS Computational Biology, 2, 1–4, https://doi.org/10.1371/journal.pcbi.0020087, 2006.

Boehm, B. W.: A Spiral Model of Software Development and Enhancement, ACM SIGSOFT Software Engineering Notes, 11, 14–24, https://doi.org/10.1145/358886.358895, 1986.

Boulanger, A.: Open-source versus proprietary software: Is one more reliable and secure than other?, IBM Systems Journal, 1, 239–248, https://doi.org/10.1147/sj.442.0239, 2005.

Carlson, D., Eyring, V., van der Wel, N., and Langendijk, G.: WCRP's Coupled Model Intercomparison Project: A Remarkable Contribution to Climate Science https://www.wcrp-climate.org/images/modelling/WGCM/CMIP/CMIP6FinalDesign_GMD_180329.pdf (último acceso 31/05/2024), 2018.

CMIP: Coupled Model Intercomparison Project, http://cmip-pcmdi.llnl.gov/ (último acceso 31/05/2024), 2025a.

CMIP: History of CMIP https://pcmdi.llnl.gov/mips/cmip5/history.html (último acceso 31/05/2024), 2025b.

Costa, L., Barbosa, S., and Cunha, J.: Evaluating Tools for Enhancing Reproducibility in Computational Scientific Experiments, in: Proceedings of the 2nd ACM Conference on Reproducibility and Replicability, ACM REP '24, p. 46–51, Association for Computing Machinery, New York, NY, USA, ISBN 9798400705304, https://doi.org/10.1145/3641525.3663623, 2024.

DoD CIO: Clarifying Guidance Regarding Open Source Software (OSS), Tech. rep., 6 pp. https://dodcio.defense.gov/Portals/0/Documents/FOSS/2009OSS.pdf (último acceso: 14/04/2024), 2009.

Drake, J. B.: Climate Modeling for Scientists and Engineers, Society for Industrial and Applied Mathematics, ISBN 978-0898714778, 2001.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Durack, P. J., Taylor, K. E., Gleckler, P. J., Meehl, G. A., Lawrence, B. N., Covey, C., Stouffer, R. J., Levavasseur, G., Ben-Nasser, A., Denvil, S., Stockhause, M., Gregory, J. M., Juckes, M., Ames, S. K., Antonio, F., Bader, D. C., Dunne, J. P., Ellis, D., Eyring, V., Fiore, S. L., Joussaume, S., Kershaw, P., Lamarque, J.-F., Lautenschlager, M., Lee, J., Mauzey, C. F., Mizielinski, M., Nassisi, P., Nuzzo, A., O'Rourke, E., Painter, J., Potter, G. L., Rodriguez, S., and Williams, D. N.: The Coupled Model Intercomparison Project (CMIP): Reviewing project history, evolution, infrastructure and implementation, EGUsphere, 2025, 1–74, https://doi.org/10.5194/egusphere-2024-3729, 2025.

Easterbrook, S. M.: Getting the source code for climate models https://www.easterbrook.ca/steve/2009/06/getting-the-source-code-for-climate-models (último acceso 31/05/2024), 2009.

Easterbrook, S. M.: Climate Change: A Grand Software Challenge, in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, FoSER '10, p. 99–104, Association for Computing Machinery, New York, NY, USA, ISBN 9781450304276, https://doi.org/10.1145/1882362.1882383, 2010.

Eyring, V., Bock, L., Lauer, A., Righi, M., Schlund, M., Andela, B., Arnone, E., Bellprat, O., Brötz, B., Caron, L.-P., Carvalhais, N., Cionni, I., Cortesi, N., Crezee, B., Davin, E., Davini, P., Debeire, K., de Mora, L., Deser, C., Docquier, D., Earnshaw, P., Ehbrecht, C., Gier, B., Gonzalez-Reviriego, N., Goodman, P., Hagemann, S., Hardiman, S., Hassler, B., Hunter, A., Kadow, C., Kindermann, S., Koirala, S., Koldunov, N., Lejeune, Q., Lembo, V., Lovato, T., Lucarini, V., Massonnet, F., Müller, B., Pandde, A., Pérez-Zanón, N., Phillips, A., Predoi, V., Russell, J., Sellar, A., Serva, F., Stacke, T., Swaminathan, R., Torralba, V., Vegas-Regidor, J., von Hardenberg, J., Weigel, K., Zimmermann, K., and Zscheischler, J.: ESMValTool (v2.0) - a community diagnostic and performance metrics tool for routine evaluation of Earth system models in CMIP, Geoscientific Model Development, 13, 1179–1199, https://doi.org/10.5194/gmd-13-1179-2020, 2020.

Fenton, N. E. and Bieman, J.: Software Metrics: A Rigorous and Practical Approach, CRC Press, 2014.

García-Rodríguez, M.: FortranAnalyser, https://zenodo.org/records/5942943, (último acceso 14/08/2025), https://doi.org/10.5281/zenodo.5942943, 2022.

García-Rodríguez, M., Añel, J. A., and Rodeiro-Iglesias, J.: Assessing and improving the quality of Fortran code in scientific software: FortranAnalyser, Software Impacts, 21, 100 692, 2024.

Gardner, J. and GitHub staff: Linguist: Language Savvy, https://github.com/github/linguist (último acceso 31/05/2024), 2024.

Geosc. Model Dev. Editors: Editorial: The publication of geoscientific model developments v1.1, Geoscientific Model Development, 8, 3487–3495, https://doi.org/10.5194/gmd-8-3487-2015, 2015.

Hunter-Zinck, H., de Siqueira, A. F., V´squez, V. N., Barnes, R., and Martinez, C. C.: Ten simple rules on writing clean and reliable open-source scientific software, PLOS Computational Biology, 17, 1–9, https://doi.org/10.1371/journal.pcbi.1009481, 2021.

Kanewala, U. and Bieman, J. M.: Testing scientific software: A systematic literature review, Information and Software Technology, 56, 1219–1232, https://doi.org/10.1016/j.infsof.2014.05.006, 2014a.

Kanewala, U. and Bieman, J. M.: Testing scientific software: A systematic literature review, Information and Software Technology, 56, 1219–1232, https://doi.org/10.1016/j.infsof.2014.05.006, 2014b.

Kernighan, B. W. and Ritchie, D. M.: The C Programming Language, Prentice Hall, ISBN 978-0131103627, 1988.

Knutti, R., Masson, D., and Gettelman, A.: Climate model genealogy: Generation CMIP5 and how we got there, Geophysical Research Letters, 40, 1194–1199, https://doi.org/10.1002/grl.50256, 2013.

Matthes, E.: Python Crash Course, No Starch Press, ISBN 978-1593279288, 2019.

McCall, J., Richards, P., and Walters, G.: Factors in Software Quality, Proceedings of the Second International Conference on Software Engineering, 2, 155–160, https://doi.org/RADC-TR-77-369, 1977.

McConnell, S.: Code Complete, Microsoft Press, 2004.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

Metcalf, M., Reid, J., and Cohen, M.: Modern Fortran Explained, Oxford University Press, numerical recipes in fortran 90 edn., ISBN 978-0199601424, 2011.

Moore, G. E.: Cramming more components onto integrated circuits, McGraw-Hill, 38 edn., 1965.

Nature: Does your code stand up to scrutiny?, Nature, 555, https://doi.org/10.1038/d41586-018-02741-4, 2018.

Nguyen-Hoan, L., Flint, S., and Sankaranarayana, R.: A Survey of Scientific Software Development, in: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '10, Association for Computing Machinery, New York, NY, USA, ISBN 9781450300391, https://doi.org/10.1145/1852786.1852802, 2010.

Perkel, J. M.: Containers in the Cloud, Nature, SIAM News, 1, 247–248, https://doi.org/10.1038/d41586-019-03366-x, 2019.

Pipitone, J. and Easterbrook, S.: Assessing climate model software quality: a defect density analysis of three models, Geoscientific Model Development, 5, 1009–1022, https://doi.org/10.5194/gmd-5-1009-2012, 2012.

Pressman, R. S.: Software Engineering: A Practitioner's Approach, McGraw-Hill, 2005.

RealClimate.org: Sources of code and data related to climate science http://www.realclimate.org/index.php/data-sources (último acceso 31/052024), 2009.

Riesch, M., Nguyen, T. D., and Jirauschek, C.: bertha: Project skeleton for scientific software, PLOS ONE, 15, 1–12, https://doi.org/10.1371/journal.pone.0230557, 2020.

Seibel, P.: Practical Common Lisp, Apress, Berkeley, CA, ISBN 978-1-59059-239-7, https://gigamonkeys.com/book/, 2005.

Stodden, V., Guo, P., and Ma, Z.: Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals, PLoS ONE, 8, e67 111, https://doi.org/10.1371/journal.pone.0067111, 2013.

Stodden, V., Seiler, J., and Ma, Z.: An empirical analysis of journal policy effectiveness for computational reproducibility, Proceedings of the National Academy of Sciences, 115, 2584–2589, https://doi.org/10.1073/pnas.1708290115, 2018.

Stouffer, R., Eyring, V., Meehl, G., Bony, S., Senior, C., Stevens, B., and Taylor, K.: CMIP5 Scientific Gaps and Recommendations for CMIP6, Bulletin of the American Meteorological Society, 98, 95–105, https://doi.org/10.1175/BAMS-D-15-00013.1, 2017.

Stroustrup, B.: The C++ Programming Language, Addison-Wesley, 4th edn., ISBN 978-0321563842, 2013.

Trisovic, A., Lau, M. K., Pasquier, T., et al.: A large-scale study on research code quality and execution, Scientific Data, 9, 60, https://doi.org/10.1038/s41597-022-01143-6, 2022.

U.S. Code: 17 U.S. Code § Section 105. Subject matter of copyright: United States Government works, 1976.

van Wendel de Joode, R., de Bruijn, J. A., and van Eeten, M. J. G.: Protecting the Virtual Commons, T.M.C. Asser Press, ISBN 978-90-6704-159-1, 2003.

Wieters, N. and Fritzsch, B.: Opportunities and limitations of software project management in geoscience and climate modelling, Advances in Geosciences, 45, 383–387, https://doi.org/10.5194/adgeo-45-383-2018, 2018.

Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., Haddock, S. H. D., Huff, K. D., Mitchell, I. M., Plumbley, M. D., Waugh, B., White, E. P., and Wilson, P.: Best Practices for Scientific Computing, PLOS Biology, 12, 1–7, https://doi.org/10.1371/journal.pbio.1001745, 2014.

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., and Teal, T.: Good enough practices in scientific computing, PLOS Computational Biology, 13, e1005 510, https://doi.org/10.1371/journal.pcbi.1005510, 2017.

Wu, C., Chakravorti, T., Carroll, J. M., and Rajtmajer, S.: Integrating measures of replicability into scholarly search: Challenges and opportunities, in: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems, CHI '24, Association for Computing Machinery, New York, NY, USA, ISBN 9798400703300, https://doi.org/10.1145/3613904.3643043, 2024.

https://doi.org/10.5194/egusphere-2025-6108
Preprint. Discussion started: 12 March 2026

410 Zaporozhets, D.: GitLab, https://about.gitlab.com/ (último acceso 31/05/2024), 2011.