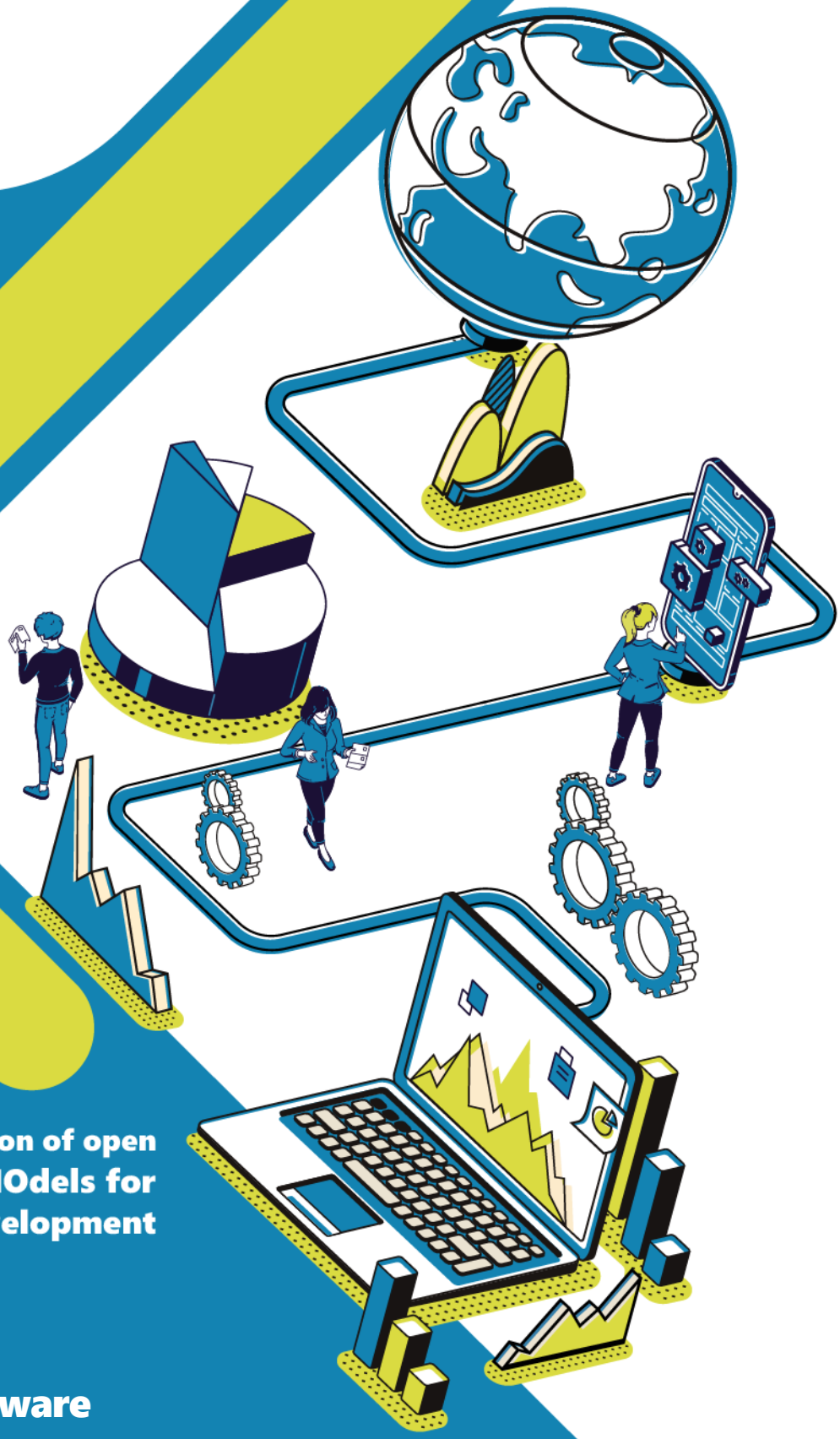
DIAMOND

www.climate-diamond.eu



Delivering the next generation of open Integrated Assessment MOdels for Net-zero, sustainable Development

D6.6 Open-source software development pipelines

WP6 – Open

30/05/2025

Funded by the European Union

Disclaimer

Copyright Message

Grant Agreement Number	**101081179**		**Acronym**	**DIAMOND**
Full Title	Delivering the next generation of open Integrated Assessment MOdels for Net-zero, sustainable Development			
Topic	HORIZON-CL5-2022-D1-02			
Funding scheme	HORIZON EUROPE, RIA – Research and Innovation Action			
Start Date	December 2022	**Duration**	48 Months	
Project URL	http://www.climate-diamond.eu/			
EU Project Advisor	Silvia Vaghi			
Project Coordinator	Institute of Communications and Computer Systems - ICCS			
Deliverable	D6.6 – Open-source software development pipelines			
Work Package	WP6 – Open			
Date of Delivery	**Contractual**	31/05/2025	**Actual**	30/05/2025
Nature	Report	**Dissemination**	Public	
Lead Beneficiary	Institute of Communications and Computer Systems - ICCS			
Responsible Authors	Konstantinos Koasidis (ICCS)	**Email**	kkoasidis@epu.ntua.gr	
	Georgios Xexakis (HOL)	**Email**	gxexakis@holisticsa.gr	
Contributors	Alexandros Nikas (ICCS)			
Reviewer(s)	Jan Ivar Korsbakken (CICERO), Clàudia Rodés Bachs (BC3)			
Keywords	development pipelines; open-source software; data management; versioning; continuous integration; evaluation; refactoring			

EC Summary Requirements

1. Changes with respect to the DoA

No changes with respect to the work described in the DoA.

2. Dissemination and uptake

The deliverable will serve as a guide for IAM modelling teams to get familiar with modern software engineering practices, identify their usefulness in terms of model development, and adopt them in their work. The main audience for the deliverable will be the DIAMOND modelling teams and the general modelling community beyond the project.

3. Short summary of results (<250 words)

This report synthesises and contextualises common and pertinent software engineering practices with the potential to improve the efficiency, transparency, and credibility of IAM development and use. The practices are collected from the scientific literature on software engineering and are adapted to the needs and requirements of IAM development. Compared to previous efforts that aimed to synthesise best practices for model development and use, we instead develop a comprehensive framework of state-of-the-art practices that does not only provide a useful checklist for modellers but also hands-on examples on how to implement these practices. Contrary to model use, which was addressed to a larger extent in the recent literature—e.g., in terms of harmonisation techniques or participatory modelling practices—our framework focuses on aspects of model development instead, which has so far remained largely unexplored. After its elaboration, the framework is used to evaluate the current practices of the modelling teams of DIAMOND based on a survey, which was distributed to the modelling teams between April and May 2025 and included questions aiming to identify their priorities and potential areas of improvement. Along with additional reports on open science, the framework presented here will be operationalised as model development progresses.

4. Evidence of accomplishment

This report and the results of the survey (including the scripts for the analysis) versioned in Zenodo[1].

[1] https://doi.org/10.5281/zenodo.15548277

Preface

DIAMOND will update, upgrade, and fully open six Integrated Assessment Models (IAMs) that are emblematic in scientific and policy processes, improving their sectoral and technological detail, spatiotemporal resolution, and geographic granularity. It will further enhance modelling capacity to assess the feasibility and desirability of Paris-compliant mitigation pathways, their interplay with adaptation, circular economy, and other SDGs, their distributional and equity effects, and their resilience to extremes, as well as robust risk management and investment strategies. This will be done via integration of tools and insights from psychology, finance research, behavioural and labour economics, operational research, and physical science. The project will develop a transdisciplinary scientific approach to legitimise the implementation process and co-create research questions that stretch the frontiers of climate science, as well as establish vibrant communities of practice to transparently open model enhancements and to develop capacities, thereby lowering the entrance barriers to the established IAM community.

ICCS	INSTITUTE OF COMMUNICATIONS AND COMPUTER SYSTEMS	EL
BC3	ASOCIACION BC3 BASQUE CENTRE FOR CLIMATE CHANGE - KLIMA ALDAKETA IKERGAI	ES
CESAR	KRATENA KURT	AT
CICERO	CICERO SENTER FOR KLIMAFORSKNING	NO
CYI	THE CYPRUS INSTITUTE	CY
E4SMA	ENERGY ENGINEERING ECONOMIC ENVIRONMENT SYSTEMS MODELING AND ANALYSIS SRL	IT
HOLISTIC	HOLISTIC IKE	EL
COMILLAS	UNIVERSIDAD PONTIFICIA COMILLAS	ES
ISINNOVA	ISTITUTO DI STUDI PER L'INTEGRAZIONE DEI SISTEMI (I.S.I.S) - SOCIETA'COOPERATIVA	IT
SEURECO	SEURECO SOCIETE EUROPEENNE D'ECONOMIE SARL	FR
UM	UNIVERSITEIT MAASTRICHT	NL
ESMIA	ESMIA CONSULTANTS INC.	CA
USMF	THE UNIVERSITY OF MARYLAND FOUNDATIION INC	US
UMD	UNIVERSITY OF MARYLAND	US
EPFL	ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE	CH
ETH	EIDGENOESSISCHE TECHNISCHE HOCHSCHULE ZUERICH	CH
UNIBAS	UNIVERSITAT BASEL	CH
Imperial	IMPERIAL COLLEGE OF SCIENCE TECHNOLOGY AND MEDICINE	UK
Oxford	THE CHANCELLOR, MASTERS AND SCHOLARS OF THE UNIVERSITY OF OXFORD	UK
UCL	UNIVERSITY COLLEGE LONDON	UK

Executive Summary

With their complexity and sophistication, integrated assessment models (IAMs) are often considered black boxes, prompting calls for improving their transparency and reproducibility. Past efforts towards these goals include opening up models and creating vibrant communities around them, formalising methods for evaluating modelling results, establishing consistent result templates, and improving data management. While many of these activities are inspired by good practices in the software engineering (SE) community, other practices have been less widespread in model development. Despite the ubiquity of agile software development in the industry, user-centric methods are rare in IAMs and relevant models (e.g., energy system models), as their development is often driven by modellers based on rigid planning rather than on frequent stakeholder interactions. Similarly, other established SE practices are not the norm in IAM development such as automated code testing or API methods for data exchange, despite many proposed methodologies for model linking. This study aims to synthesise pertinent SE practices with the potential to improve the efficiency, transparency, and credibility of model development, especially on IAMs. The practices are collected from the literature on scientific SE and are adapted to modelling needs and requirements, building on previous efforts to distil best practices.

Overall, 18 practices have been identified and categorised into five phases within the lifecycle of IAM development. The first phase includes data processing methods (e.g., automated processing scripts under version control), methods for eliciting stakeholder requirements and refactoring model code, procedures for managing the lifecycle of development from project conception to delivery (e.g., agile development), as well as programming paradigms (e.g., object-oriented programming), design patterns (e.g., Iterators in Python), and coding standards (e.g., PEP 8 in Python). The next phase includes various types of documentation used in modelling projects (e.g., conceptual descriptions of models, code comments, metadata, and tutorials), while the evaluation phase includes code testing techniques (e.g., unit testing), processes for continuous integration (e.g., automated code testing), and methods for model validation (e.g., historical simulations). The next phase includes practices related to version control and issue tracking to support collaborative development as well as practices related to open-source development (e.g., proper licensing). Finally, we identify practices on the documentation, management, and automation of third-party dependencies and system requirements as well as practices related to the so-called continuous deployment—i.e., automation processes to reduce the time between development and release of a new software version.

We then perform a preliminary evaluation of the current use of these practices in the modelling community based on a survey with nine modelling teams in DIAMOND. Each respondent represented their whole team and, in most cases, a single model. The respondents read a short definition for each of the 18 practices and provided their feedback on whether they have used (or are currently using) any of them and whether they think they are relevant for the development and uptake of their model. Most practices related to documentation, versioning, and open access were considered as relevant or very relevant by the responding teams, along with practices related to data processing and model validation. These results correspond to requirements that have been frequently emphasised in the literature. However, the reported practices that are used by the teams on these topics are not always aligned with best practices. For instance, almost all teams document input datasets using direct comments or external README files instead of structured metadata standards. In addition, half of the models use datasets that are not open access while alternative open datasets are not readily available. In terms of evaluation, most teams have performed sensitivity analysis on their models but do not yet have experience with the vetting process followed in the Intergovernmental Panel on Climate Change (IPCC) scenario processes of the 6th Assessment Report (AR6), despite its potential reuse in the AR7 cycle.

Other practices were deemed as less relevant for IAM development, including some of the most popular practices in SE, such as development lifecycle models, continuous integration, and continuous deployment. Most surveyed

teams appear not to have used agile methods or any explicit lifecycle model to guide their development, established automated systems for testing or deploying, or tested individual functions and routines. Since barely any team has members with SE background, it is understandable that they are hesitant to invest time and effort to implement such practices—especially when they may also lack the necessary know-how to do so effectively or the positive impacts are not immediately clear. As such, practices perceived as of low relevance may still be extremely important to pursue (i.e., through standardised processes and training) to exploit untapped potential and improve the long-term sustainability and usability of the models.

While these results are not necessarily representative of the wider modelling community, they indicate that current IAM development pipelines could be further improved based on good SE practices and that entry points may be needed for practices that modellers are still sceptical about. A follow-up of this study aims to clarify the benefits of using modern SE methods for IAM development and identify easy ways to implement them, starting from the IAMs participating in the DIAMOND project. Both the survey and all follow-up tutorials, checklists, and recommendations will be adapted and distributed to the broader modelling community of practice in order to support the adoption of good development practices that can improve collaboration among, comprehensibility of, and trust in IAMs and relevant models.

Funded by
the European Union

Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Context

Integrated assessment models (IAMs) have played an increasingly important role in the last two decades in guiding climate action, helping design ambitious targets, and underpinning scientific assessments on climate change such as the Assessment Reports (AR) of the Intergovernmental Panel for Climate Change (IPCC) (Fisher-Vanden & Weyant, 2020; Plazas-Niño et al., 2025). IAMs have been also used in many other applications such as on designing and assessing COVID responses, addressing geopolitical energy questions, and addressing equity in different levels (Dekker et al., 2024). Still, in a world living recently through a polycrisis, including a health emergency, an energy crisis, geopolitical conflicts, and, recently, an unfolding trade war, rapid assessments are often required beyond the typical policy cycles. Thus, modelling responses need to be responsive, transparent, and trustworthy in order to meaningfully support strategic policy design.

1.2 Challenges in IAMs

With their complexity and sophistication, IAMs are often considered as "black boxes", despite calls and actions from the modelling community for improving their transparency (DeCarolis et al., 2012; Keppo et al., 2021; Krawczyk & Braun, 2025; Pfenninger et al., 2018; Skea et al., 2021). Increasingly, more IAMs and energy models release their model code on GitHub and other code repositories, such as the GCAM model, OSeMOSYS-CLEWs, PyPSA, Calliope and MESSAGE-ix (Bond-Lamberty et al., 2019; Huppmann et al., 2019; Pfenninger et al., 2017, 2018; Plazas-Niño et al., 2025). This is often leading to vibrant communities of practice as more and more researchers and practitioners can take up an open model. However, transparency in model code does not always translate into transparency in data. Many datasets that are frequently used by IAMs are not open access, such as proprietary datasets from the International Energy Agency (IEA) on energy supply and demand and the recent Global Trade Analysis Project (GTAP) databases on trade patterns. Developing high-quality datasets is resource- and time-consuming and cannot be always provided free of charge. Yet, alternative datasets could be also suggested by open-source models.

In addition, code or data transparency does not imply comprehensibility. Without adequate documentation, it is often challenging to understand the conceptual design of a model and how its various systems are interacting with each other (DeCarolis et al., 2012; Robertson, 2021). Many models are now documented in some form, e.g., in the IAMC wiki[2], the IAM PARIS platform[3], or model-specific repositories on GitHub[4], but the current content of this documentation may not be enough. As even experts are understanding and using modelling results in different ways (Braunreiter & Blumer, 2018), it is critical that modelling documentation promote trust and understanding to all relevant stakeholders, including other modelling teams, non-modelling researchers, and, especially, decision makers and other users of modelling results. The comprehensibility goal could be further promoted by enabling more opportunities for stakeholder interactions within the actual process of model development (McGookin et al., 2024), helping stakeholders to better conceptualise the model. This could also improve the match between envisioned model functionalities and users' needs, something that may be lacking at times (Süsser et al., 2022).

Apart from enhancing model transparency, it is crucial to ensure the quality and validity of modelling results. There

[2] https://www.iamcdocumentation.eu/index.php/IAMC_wiki
[3] https://iamparis.eu/detailed_model_doc
[4] e.g., the GCAM GitHub repository (https://github.com/JGCRI/gcam-core)

has been much progress towards these goals with the formalisation of methods for evaluating modelling results (C. Wilson et al., 2021), the release of multiple diagnostic tools to understand model behaviour and interpret results (Harmsen et al., 2021), and the mainstreaming of open science principles in influential modelling communities such as the Integrated Assessment Modelling Consortium (IAMC) (Skea et al., 2021). Still, there is much room for improvement, for instance, on the vetting method that has been used for data curation and results validation in the IPCC AR6 scenario database, which may have, inadvertently, reinforced biases in the scenario ensemble (Peters et al., 2023). Similarly, there is potential for improving model intercomparison, a key validation method where various models solve for the same scenario protocol (C. Wilson et al., 2021). Despite assumed as a "weak form of validation", since model results are not compared with real world observations (DeCarolis et al., 2012), intercomparisons are popular as they allow to evaluate the robustness of modelling results under various modelling paradigms and methodologies. Nevertheless, due to the variability in the architecture of the different models, the harmonisation of input data can be challenging (Giarola et al., 2021), while intercomparison protocols may not always be clearly defined or structured as they could be (Mikropoulos et al., 2025; Nikas et al., 2021; Peters et al., 2023).

Even by achieving adequate transparency and validation in models and modelling exercises, the often-sought goal of reproducibility can be difficult to attain (DeCarolis et al., 2012; Robertson, 2021). IAMs can be notoriously challenging to run, as, depending on their complexity and solution algorithm, they can easily become very resource-intensive, requiring powerful computers. In addition, models may have multiple dependencies, ranging from libraries that need to be installed prior to running to specific operating systems and configurations, further complicating their use by third parties. And while some models include user interfaces for enabling smooth data input or visualisations of results (Huppmann et al., 2019; Rodés-Bachs et al., 2024), they are often the exception as several models require extensive data processing activities to run (Bond-Lamberty et al., 2019), while their interfaces are not designed with non-modeller users in mind. All these issues may further complicate the reproducibility of a model, as any third-party model tester would need to have both the time and resources to install and run the model, on top of having access to all model code and data and the basic capacity to understand what the model does.

1.3 Modern software engineering practices

In parallel with the advancement of IAMs in the last decades, the overall discipline of software engineering (SE) was witnessing its own breakthroughs. Instead of the traditional top-down planning approaches ("waterfall" model), "agile" methodologies have been increasingly used for the management of software development projects, promoting collaborative development, minimising unnecessary work, and accepting uncertainty as an integral part of software development process (Dingsøyr et al., 2012). Most importantly, agile methodologies emphasise continuous and meaningful interactions with clients and stakeholders, bringing increased efficiency and stakeholder satisfaction (Dybå & Dingsøyr, 2008; Serrador & Pinto, 2015). Apart from improving methodologies for eliciting software requirements—the so-called requirements engineering (Curcio et al., 2018)—the emphasis on frequent user interactions during software development led to increased significance of operational aspects such as software deployment and quality assurance processes. Thus, several "continuous" SE practices have been introduced and adopted by the industry, including continuous testing, integration, delivery, and deployment (Fitzgerald & Stol, 2017). While the implementation of these practices is not without its challenges, it can provide many benefits to software projects (Soares et al., 2022), such as automating testing processes, facilitating deployment, detecting errors and increasing security, enhancing scalability, and overall improving reliability (Shahin et al., 2017).

Through interactions between industry and science, these modern software development practices found their

Funded by
the European Union

way to scientific software development. Based on a systematic literature review, Heaton and Carver (2015) analysed multiple claims on the use of such practices in scientific software development and found a mostly positive impact on the quality of the software and the efficiency of its development. Nevertheless, testing practices, requirements elicitation, and project management methods that are typically used in the industry are less commonplace in scientific software, often due to the lack of formal SE training and the difficulty of validating scientific software in general (Heaton & Carver, 2015). At the same time, scientific software development does not always have a clear "client" to elicit requirements from (e.g., a funder) and can follow a more explorative approach. While there are many similarities between scientific and business-oriented software development, e.g., the need for structured code testing, many industry-inspired practices need to be adapted to the requirements and the processes of scientific software. Mindful of such challenges and the general need for scientific processes to be reproducible, Sandve et al. (2013) outlined ten basic rules for computational research, recommending the use of version control, the documentation of scripts and intermediate results, and the open sharing of all code, data, and results. Similarly, Wilson et al. (2017) suggested good enough practices for scientific computing that can be easier to adopt than the optimal "best practices", including advice on creating and documenting analysis-friendly data, making software dependencies and requirements explicit, and keeping changes in the software short as well as sharing them frequently.

1.4 Software engineering practices in IAMs

Some of these practices also found their way to the modelling world, especially those related to open access and data management. For instance, the release of open-source models such as GCAM (Bond-Lamberty et al., 2019; Rodés-Bachs et al., 2024) and OSeMOSYS (Plazas-Niño et al., 2025) created vibrant research communities around them that promoted their open-source paradigm. This was further emphasised by multiple calls in the literature that promoted the value of open-source modelling and operationalised its principles (Pfenninger et al., 2014, 2018), the establishment of relevant initiatives such as the Open Energy Modelling Initiative, and the overall open access movement that has transformed the publishing and research world (Demeter et al., 2021). Similarly, the established data management principles of findability, accessibility, interoperability, and reusability (FAIR) (Wilkinson et al., 2016) have been recently applied to IPCC processes (Stockhause et al., 2024) and in general to modelling practices. Furthermore, the establishment of a consistent template by the IAMC has helped to standardize IAM results, including in IPCC assessment reports (Gidden & Huppmann, 2019), along with the development of Python-based applications such as pyam[5] and nomenclature[6] that have enabled data manipulation and validation based on these templates.

Other modern SE practices are less widespread in IAM development. Despite the ubiquity of formal agile methodologies in industry and calls in the IAM literature to adopt them (Knapen et al., 2010), they are not typically used in the development of IAMs or, in general, in the domain of scientific software development (Heaton & Carver, 2015). Similarly, efforts to use established development paradigms and design patterns in modelling frameworks have been only scarcely used (David et al., 2013). In addition, while many practices have been recommended for the validation of modelling results (C. Wilson et al., 2021) and for ensuring their impact in the science-society interface (Hamilton et al., 2019), automated code testing practices (e.g., unit tests) are not the norm in IAM development (Iwanaga et al., 2018). Apart from reinforcing the credibility of the modelling results, these tests can be very useful for ensuring quality assurance when the model code needs to change, e.g. adding new features to the model. Thus, without tests, every new model version needs to be validated quasi-manually both for the new and existing functionalities, increasing developing time and potentially affecting quality. Similar

[5] https://github.com/iamconsortium/pyam
[6] https://github.com/IAMconsortium/nomenclature

Funded by the European Union

challenges exist in model linking applications. While several approaches and frameworks have been suggested for linking IAMs with other models (Belete et al., 2017; Hinkel, 2009; Huppmann et al., 2019; Siebers et al., 2020; Warren et al., 2008), none of them has been established in the IAM community in the way that, e.g., REST APIs, became one of the most widely used standards for web applications to talk to each other.

1.5 Goal of the study

Based on the notion that IAM development can benefit from recent advancements in software engineering, this study aims to synthesise pertinent SE practices with the potential to improve the efficiency, transparency, and credibility of the development and use of IAMs. The practices are collected from the literature on scientific and general SE and are adapted to the needs and requirements of IAM development. Compared to previous efforts that aimed to synthesise best practices for model development of use (DeCarolis et al., 2012, 2017; Jakeman et al., 2006), this study aims to develop a comprehensive framework of state-of-the-art practices that does not only provide a useful checklist for modellers but also hands-on examples on how to implement these practices. After its elaboration in Section 2, the framework is used to evaluate the current practices of the modelling teams of DIAMOND, identifying their priorities and potential areas of improvements (Section 3). Along with open-science protocols as part of the broader activities of WP6, work on this task will continue with the operationalisation of this framework as model development progresses (see also Section 4 for next steps).

2 Software Engineering Practices in IAMs

An extensive literature review was performed, focusing on software engineering practices that are used in or suggested for the development of integrated assessment, energy, or environmental models (David et al., 2013; DeCarolis et al., 2012; Huppmann et al., 2019; Knapen et al., 2010; Pfenninger et al., 2018; C. Wilson et al., 2021). This review was complemented with literature on scientific software development in general (Heaton & Carver, 2015; Sandve et al., 2013; G. Wilson et al., 2017), aiming to find relevant practices that could be implemented in model development and evidence of their effectiveness in other fields. Overall, eighteen practices were collected and have been categorised by the authors in a framework of five critical phases in the lifecycle of IAMs, including model development, documentation, evaluation, versioning and collaboration, and deployment. The framework focuses on aspects of model development rather than on model use, since the latter was much more covered in the recent literature, e.g. in terms of harmonisation techniques (Giarola et al., 2021; Rodés-Bachs et al., 2025) or participatory modelling practices (McGookin et al., 2024). The framework is shown in Table 1, and all practices are presented in the rest of the chapter.

Table 1. Software engineering practices for five different phases of IAM development and release

Model development	Documentation	Evaluation	Versioning and collaboration	Deployment
Requirements Engineering	Conceptual documentation	Code testing	Version control	Dependency management
Practices for data processing	Code documentation	Model validation	Issue tracking	System requirements management
Programming paradigms, patterns, and standards	Data documentation	Continuous integration	Open-source development	Continuous deployment
Development lifecycle models	Tutorials and case studies			
Practices for refactoring and adding new features				

2.1 Model development

2.1.1 Requirements engineering

Requirements engineering is the process of identifying, eliciting, and managing the needs and ideas of different stakeholders for the development of a software (Pohl, 2025). The process can be found in many different variations across the literature and practice, but it usually includes methods to elicit feedback such as interviews, workshops, and surveys along with various methods for contacting feasibility analyses and translating requirements into software specifications. In the context of IAM development—and DIAMOND in particular—requirements are elicited directly from the grant description, the engagements in WP2, from other modelling teams, and from the literature.

2.1.2 Practices for data processing

Data processing is an umbrella term for various processes to make data usable by a specific application and includes various steps such as data collection, cleaning, and transformation (Batini & Scannapieco, 2016). There are several practices for improving data and information quality, including the FAIR principles for ensuring findable, accessible, interoperable, and reusable data (Wilkinson et al., 2016) and the TRUST principles for transparent, responsible, user-focused, sustainable, and technologically-capable data repositories (Lin et al., 2020).

Data processing plays an important role in IAM research based on the vast amount of input data that most IAMs require. Good practices include automating data processing through scripts (G. Wilson et al., 2017) and using data quality assurances such as the pedigree matrix approach used in life-cycle assessment (LCA) research (DeCarolis et al., 2017). In case that scripts are not used for data processing, it is suggested to at least provide links to the original datasets and describe in a README file all the data processing steps that are taken (G. Wilson et al., 2017). Another good practice is using version control to monitor changes in the data and the processing scripts (Heaton & Carver, 2015). While version control platforms such as GitHub can be also used, they may not be ideal for large datasets. For instance, GitHub has a limitation of 100 MB per file[7], recommending the use of other types of storage such as the GitHub Large File Storage (LFS) service. Other options for storing large datasets include Zenodo (up to 50 GB) or database servers and services such as the ones provided by Microsoft Azure[8] or Amazon Web Services[9].

2.1.3 Programming paradigms, patterns, and standards

Programming paradigms, design patterns, and coding standards can be used throughout the software development to improve and adapt it to the needs of the application. Firstly, programming paradigms are high-level approaches to structure a computer program. Pertinent examples include object-oriented programming where code is organised into classes of objects that interact with each other or functional programming where code is organised in a series of function evaluations, avoiding thus mutable objects. Secondly, design patterns are efficient ways to solve typical problems in software design such as using Prototypes, a design pattern that allows developers to create new objects by cloning an existing object (the prototype), reducing the computational cost and required time of creating a new object from scratch. Another prominent design pattern is an Iterator, a collection of pointers to specific objects that allow the user to traverse over these objects in a specific order (Gamma et al., 1994). Many design patterns are already integrated in programming languages (such as Iterators in Python and Prototypes in JavaScript) while others can be explicitly used by developers for a new software. Lastly, coding standards such as PEP 8[10] in Python can be used to create more usable and readable code by using coding conventions, standard layouts, effective documentation and among others.

2.1.4 Software development lifecycle models

Software development encompasses many different steps from project conception to delivery. Various models exist to manage these steps along the full software development lifecycle such as the waterfall, the iterative, and the agile model (Knapen et al., 2010). In the waterfall model, the lifecycle is followed in a linear fashion and each phase proceeds after the previous has finished. In contrast, in the iterative approach, software is created in an incremental way and based on many cycles of development and feedback by the users. Agile development is

[7] https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github
[8] https://azure.microsoft.com/en-us/products/azure-sql/database/
[9] https://aws.amazon.com/products/databases/
[10] https://peps.python.org/pep-0008/

based on the iterative model, but it is even more user-centric, based on constant feedback by the user instead of enacting a comprehensive planning at the beginning of the process.

2.1.5 Practices for refactoring and adding new features

Refactoring is the process of re-writing existing code in order to improve it, e.g., in terms of maintainability and/or flexibility. For example, breaking up big, complex functions may increase clarity and improve code readability. In addition, it can facilitate future changes in the code and the introduction of new features, while reducing the risk of bugs when doing so. There are many good practices to follow during refactoring (Fowler, 2018), such as writing regression tests and using a Test-Driven Development approach in general. In this approach, a developer writes tests in parallel with the development of new code to confirm that the intended functionalities are implemented without the introduction of bugs. Thus, the refactored code would need to succeed in the same tests as the original one to ensure its functionality. The same test-first approach can be used when adding new features to a software, to ensure that both the new and existing functionalities are working.

2.2 Documentation

2.2.1 Conceptual documentation

This practice aims to provide a comprehensive documentation of the overall purpose and specifications of a model and, potentially, a high-level description of its components, i.e., algorithms, input data, and main outputs (Jakeman et al., 2006). This documentation may also include illustrations of the whole system that is simulated, key assumptions, uncertainties, pertinent features (such as technologies or policies modelled), as well as details about the temporal, spatial, and sectoral resolution of the model. Examples of relevant fields to be documented for IAMs can be found in the IAMC wiki[11] and the IAM PARIS platform[12].

2.2.2 Code documentation

There are many different standards and practices for documenting code across various communities and languages, but it often takes the form of a short description in the beginning such as the "docstrings" used in Python according to the PEP 257 standard[13]. Using in-code comments has been also suggested in IAMs and energy models, especially as a solution to automatically generate documentation, including inputs and outputs of different functions (Huppmann et al., 2019) and making it easier to keep documentation and code in sync (DeCarolis et al., 2012). Giving functions and variables meaningful names can also provide a level of documentation and can improve comprehensibility (G. Wilson et al., 2017). Many applications can be used to generate documentation automatically such as Sphinx[14], and they can be often combined with online platforms or wiki pages for sharing documentation such as Read the Docs[15]. At the very least, documentation can be provided through README, TXT, or Word files.

2.2.3 Data documentation

This collection of practices revolves around the documentation of all data inputs and outputs, ideally through metadata standards such as Dublin Core or at least in JSON or YAML files. Other practices include using comments

[11] https://www.iamcdocumentation.eu/index.php/IAMC_wiki
[12] https://iamparis.eu/detailed_model_doc
[13] https://peps.python.org/pep-0257/
[14] https://www.sphinx-doc.org/en/master/
[15] https://docs.readthedocs.com/platform/stable/index.html

on the datasets themselves (e.g., on the top of CSV files) or external README files. All datasets should be treated based on the FAIR data principles in order to ensure that they are easily findable and accessible (e.g., upload them in a repository such as Zenodo and assign them a Digital Object Identifier or DOI), that they allow for interoperability in terms of data processing (e.g., by using a specific data format), and for promoting reusability (e.g., through non-restrictive licenses).

Data documentation can be a challenge in IAMs and other similar models as original datasets often need to be extensively processed to be usable by a particular model and thus documentation needs to include a lot more detail than just a reference to where the data came from and when it was prepared. Ideally, data documentation needs to provide enough information so that an external user can reproduce the exact inputs to the model from scratch. For example, the documentation should explicitly mention how input datasets are downloaded or obtained, and detail on how they were or need to be processed prior to being loaded into the model (if that is not an explicit part of the model code documentation itself). This includes trivial but crucial details such as converting currencies, base years, or weights used for derivative variables, such as the purchasing power parity conversion for calculating the gross domestic product (GDP) of a country. Providing adequate information for such processing details can greatly facilitate the assessment of results from different models in model intercomparison exercises.

2.2.4 Tutorials and case studies

Apart from documentation that can help users to understand a model, it is essential to provide resources that can help users to also run the model. Examples include installation guides and manuals, QuickStart tutorials (see example from GCAM[16]), hands-on examples of common applications, or case studies that showcase full model use.

2.3 Evaluation

2.3.1 Code testing

This category includes practices to establish that code runs successfully and produces the expected results. Common testing practices include unit testing (testing individual functions, code blocks etc.), regression testing (testing whether existing code runs normally after the introduction of a new feature), integration testing (end-to-end testing of the whole model), and acceptability testing (testing in a real-world context, including common input data and assumptions). All four practices can be used in the same software engineering project, with unit tests being the most typical practice. In fact, many developers aim to provide unit tests for most code blocks or components in order to achieve high test coverage, which can be considered a metric of code quality. Nevertheless, testing practices are not as commonplace in scientific software engineering (Heaton & Carver, 2015) and, similarly, in modelling applications (Iwanaga et al., 2018). Scientific software developers often lack the capacity to execute consistent testing practices, scientific software can be complicated, and correct results that are used to validate model code are not always known (Heaton & Carver, 2015).

2.3.2 Model validation

Apart from checking whether model code works correctly, it is crucial to evaluate how well the model can match its conceptual description and real-world observations. Multiple methods exist for IAMs such as analysing the sensitivity of different model inputs and comparing model outputs with historical and near-term observations,

[16] https://jgcri.github.io/gcam-doc/user-guide.html

stylised facts, and results of other models, especially more detailed, process-based models (C. Wilson et al., 2021). The vetting process used in the compilation of the IPCC AR6 scenario ensemble is an example of such validation[17], albeit a rather minimal one. Specifically, the AR6 vetting process evaluates modelling results in terms of historical emissions and energy production variables as well as future near-term patterns, such as ensuring that there are no net negative CO_2 emissions before 2030. The assessment is not exhaustive as it mostly tests whether 13 common variables fall in specific boundaries in specific years. Still, passing the evaluation is not trivial, with many modelling results failing to be included in the scenario ensemble of AR6 and prompting suggestions for improvements in the next assessment reports of IPCC (Peters et al., 2023).

2.3.3 Continuous integration

Continuous integration is part of a family of "continuous" practices that have been increasingly used in the recent years and aim to optimise the efficiency and speed with which software developers can release new software and features (Shahin et al., 2017). As indicated by its name, the continuous integration process aims to ensure that every change to the model code can be integrated seamlessly to the existing code repository without breaking any previous functionalities. It is mainly based on automating code testing and other types of functional evaluations using testing frameworks such as pytest[18] in Python. Apart from automation tools, it is also recommended to keep changes small and use version control systems to manage them (G. Wilson et al., 2017). Continuous integration is often accompanied with subsequent practices such as continuous deployment that aims to deploy the updated code in a "production" environment, i.e., in the server of a client or in an executable format that can be directly run by the users. More details on continuous deployment can be found in Section 2.5.3.

2.4 Versioning and collaboration

2.4.1 Version control

Using a version control system such as Git is one of the common practices for collaborative model development, ensuring that all changes by different developers can be correctly integrated to the model code. Platforms such as GitHub or GitLab can provide an easy-to-use interface to version control, along with multiple other functionalities to monitor development and share code with multiple collaborators (Pfenninger et al., 2018). Nevertheless, these platforms are mostly optimised for hosting code and small-size datasets, and they often block the upload of large datasets or executables (e.g., all files above 100 MB in GitHub). This can be a problem for IAMs and related models as large datasets are often the norm requiring other solutions for version control. For more details on this topic see Section 2.1.2 on data management practices.

2.4.2 Issue tracking

Issue tracking systems allow users to document features and functionalities that need to be implemented, monitoring their progress, and recording bugs that need to be fixed. GitHub and other similar code-sharing platforms provide solid issue tracking that can be also connected to version control. Such platforms can further support collaborative development by offering functionalities for users to discuss plans and view progress through changes in the model code, as well as allowing more than one person to view and address an issue (such as GitHub

[17] For more details on the vetting criteria, see Table 11 in Annex III of the IPCC AR6 Working Group III report, available in: https://www.ipcc.ch/report/ar6/wg3/downloads/report/IPCC_AR6_WGIII_Annex-III.pdf

[18] https://docs.pytest.org/en/stable/

issues). Online to-do list apps such as Trello are also frequently used for issue tracking, allowing multiple users to access the issues but often without providing good functionalities for discussing or cross-referencing with code changes. Finally, personal to-do apps or even paper-based systems can also be used for issue tracking but without providing proper collaboration functionalities.

2.4.3 Open-source development

Open-source development refers to a collection of practices that ensure that a software can be freely distributed, while its source code can be publicly available, allowing modifications and derived works, and minimising discrimination and availability restrictions (Open Source Initiative, 2025). Practices include using open-source licenses and publicly sharing model code through platforms such as GitHub. It is noted that there are different open-source licenses recommended for code (e.g., MIT license[19]) and data (e.g., Creative Commons licenses). Also, licenses are usually divided in two broad categories with different advantages and disadvantages: 1) copyleft licenses that stipulate that derivative works should follow the same license and 2) permissive licenses that allows modifications with minimal restrictions.

2.5 Deployment

2.5.1 Dependency management

Modern software applications often include several third-party dependencies, such as libraries for performing standard processes, databases, or other operational infrastructure. In this regard, IAMs are not an exception. Apart from using multiple third-party libraries, IAMs may require or allow for interconnections with other models. Good practices on dependency management include listing dependencies explicitly in a file (G. Wilson et al., 2017)[20], such as in a requirements.txt or pyproject.toml file for Python-based software. Other suggestions include using well-maintained and, ideally, open-source third-party libraries, and, in the case of IAMs, clearly mentioning model interconnections in the documentation and describing how these can be implemented. Dependency management can be also automated through dedicated applications and libraries, such as the Poetry[21] library in Python, the npm registry in JavaScript, and the language-agnostic Conda tool for package and environment management[22].

2.5.2 System requirements management

Depending on their architecture, complexity, algorithmic design, and solver type, IAMs can often require significant system resources, including computational power, memory, or disk space. In additional, many models may require a specific operational system to run (Windows, Mac, or Linux) or even a specific system version or system libraries (e.g., DLL libraries in Windows). These requirements need to be clearly documented to the users and, like the third-party requirements, can be also managed by different applications. For instance, the Docker platform can be used to provide the system requirements and the operating system that is needed by the model, along with all the third-party dependencies.

2.5.3 Continuous deployment

The next step after continuous integration, continuous deployment is a series of automation processes that aim to reduce the time between development and release of a new software version. Automated tests run when a

[19] https://choosealicense.com/licenses/mit/
[20] https://creativecommons.org/share-your-work/
[21] https://python-poetry.org
[22] https://anaconda.org/anaconda/conda

change is submitted to the version control system, and, if they are successful, then the change is automatically integrated to the software code (continuous integration) and also deployed to the so-called production system, for instance a server (continuous deployment). In the case of IAMs, the output from the production system can take the form of a Docker image that can be downloaded by any user and installed in any personal computer or server without any other specification of third-party dependencies.

Funded by
the European Union

3 Survey on current practices within DIAMOND

A survey on current software engineering practices was distributed to DIAMOND's modelling teams from 16 April to 6 May 2025. The main goal of the survey was to familiarise modellers with the SE practices contained in the framework of Chapter 2 and ask whether modellers are currently using these practices (and how) and whether these practices seem relevant for their work or not. This feedback will be then used to prioritise further work in Task 6.3, understanding which practices are currently underused and which are mostly needed by the modellers. Subsequently, practical tutorials, checklists, and recommendations will be developed to support the modellers and enhance the openness and comprehensibility of the models of the project.

3.1 Methodology

The primary audience was the modelling teams of Work Package 3 (WP3), who are developing the project's new open-source models, forming the main key exploitable result of the project and the focal point of the open-source software development pipelines of this deliverable. Nevertheless, the WP4 modelling partners were also invited to fill the survey as they include teams with established models, such as ENGAGE and MAgPIE, that can provide interesting insights on their modus operandi. The survey can be also easily adapted and distributed to other modelling teams in the IAM community of practice, as almost all questions have been structured in a general way and do not specifically constraint to the DIAMOND project.

The survey started with a short introduction to the goal of the survey and few questions related to the respondents' identity, their organisation, and the modelling team they are representing. The respondents then read a short definition for each of the 18 practices and provided their feedback on whether they have used (or are currently using) any of them and whether they think they are relevant for the development and uptake of their model. The survey ended with a question on other practices that they are missing from the survey and provided space to the participant to make suggestions. The survey was administered using Microsoft Forms[23], while the survey script can be found in the accompanying repository in GitHub, containing all input data and code for data analysis[24].

At this first phase, nine responses were received from DIAMOND partners (N=9), including representatives from all modelling teams of WP3 and two teams from WP4 (Table 2). Each respondent represented the whole team and a single model with only two exceptions. First, both BC3 and ICCS teams filled the survey representing the GCAM model. Since they represent teams with different practices, their answers were analysed separately apart from some cases that are more related to the model than to the team. Second, two members of the E3 Modelling team filled the survey; as most of their answers coincided, their input was merged into one. The remaining part of this chapter provides a detailed analysis of the results, along with some initial insights that were extracted. The unprocessed survey answers per model and modelling team can be found in the Appendix.

Table 2. Current practices in conceptual documentation

No.	Modelling teams	WP	Model	Code repository
1	E4SMA Srl	WP3	OMNIA	Not yet available
2	SEURECO	WP3	NEMESIS-World	Not yet available
3	BC3	WP3	GCAM-Europe	https://github.com/bc3LC-GCAMEurope/gcam-core

[23] https://forms.office.com/e/vAMEGWWjCV
[24] https://github.com/CLIMATE-DIAMOND/IAM-development-practices

4	ICCS	WP3	GCAM-Europe	https://github.com/bc3LC-GCAMEurope/gcam-core
5	EPFL - LEURE	WP3	GEMINI-E3 EU	Not yet available
6	Imperial College London & The Cyprus Institute	WP3	CLEWS-EU	https://github.com/CLIMATE-DIAMOND/EU-CLEWS
7	Comillas Pontifical University	WP4	openTEPES	https://github.com/IIT-EnergySystemModels/openTEPES/
8	UCL	WP4	ENGAGE	Not available
9	ICCS – E3 Modelling	WP3	OPEN PROM	https://github.com/e3modelling/OPEN-PROM

3.2 Model development practices used

Most of the responding modelling teams based the requirements for new modelling features on the proposal call and Grant Agreement of DIAMOND to define the requirements for new modelling features (Figure 1). Requirements were also based on ideas from the literature (6 out of 9 respondents) and from other modelling teams, within and without DIAMOND (5/9). Interestingly, only three out of eight responding teams used the stakeholder input received through the WP2 engagement to inform new modelling work. This was expected as most modelling improvements were planned at the inception or the start of the project while most stakeholder input was received well into the 2nd year of the project, and it might become more useful for the model runs that will take place towards the end of DIAMOND. At first, this can seem to be at odds with many modern requirement engineering methodologies, since they are often based in frequent iterations with the "client" and other stakeholders (Curcio et al., 2018). Nevertheless, since the main "client" of DIAMOND is the European Commission itself, it might be unclear for the modellers whether requests from stakeholders can act as actual requirements, especially when they demand significant development effort.

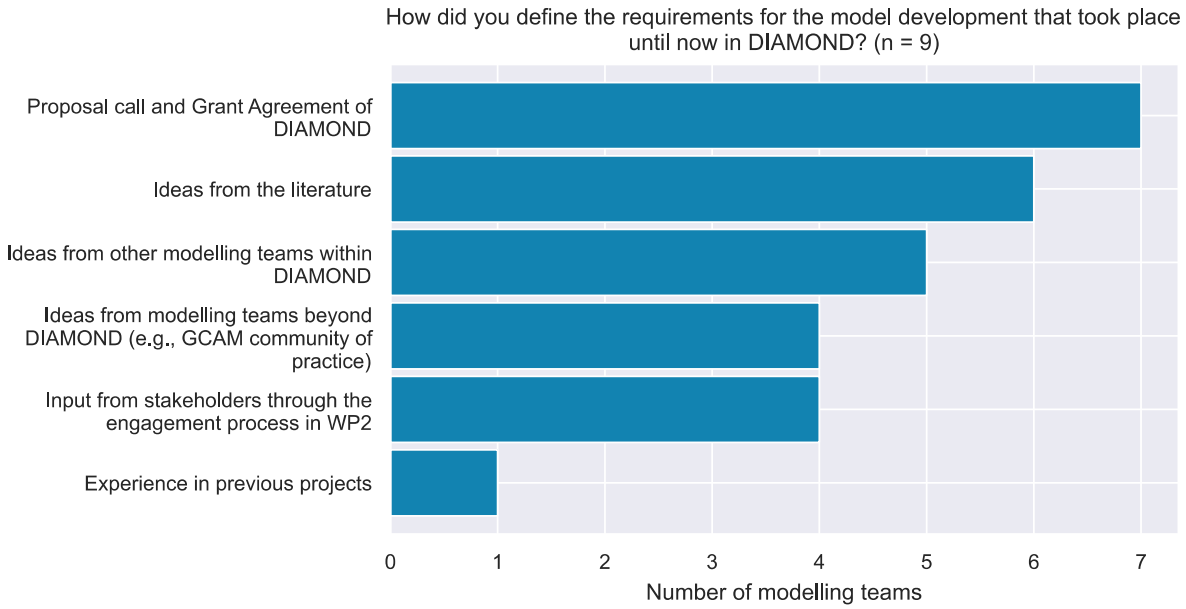


Figure 1. Current practices in requirements definition

In terms of data processing practices, most respondents are using a version control system for monitoring data changes and keep backups of both raw and processed input data (Figure 2). These practices are well aligned with recommendations and good practices for scientific computing (Sandve et al., 2013; G. Wilson et al., 2017) and IAM development (Iwanaga et al., 2018), along with the use of version controlled scripts for data processing which is

also commonly used by the respondents. Six out of the nine respondents keep the input data of their model on the same repository as their code, facilitating version control and findability (Figure 3). In addition, many teams use offline systems and other online data sharing solutions, presumably for storing datasets that are not yet open access or for very large files that normal GitHub repositories cannot store[25]. As most teams also suggest that they will eventually store data in public repositories such as Zenodo, it is assumed that these data will become openly available towards the end of the project.

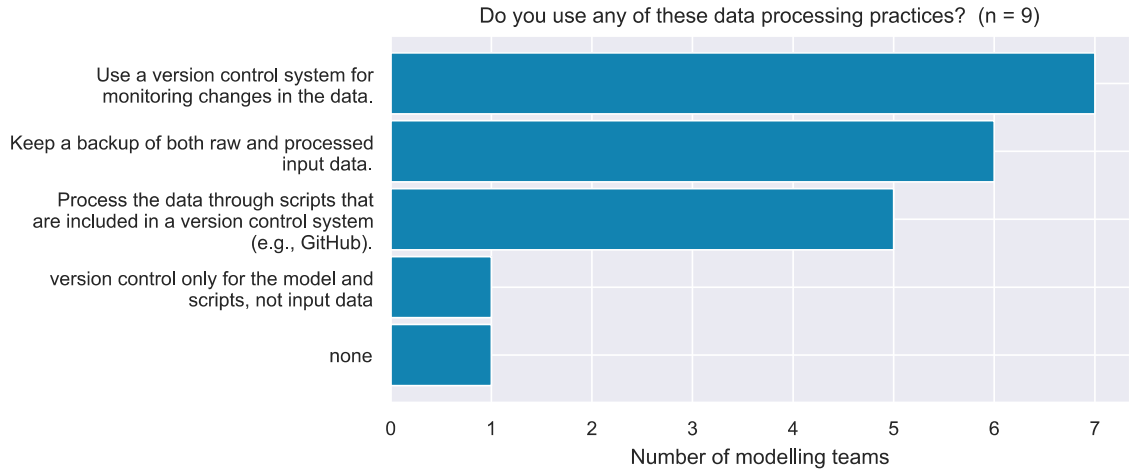


Figure 2. Current practices in data processing

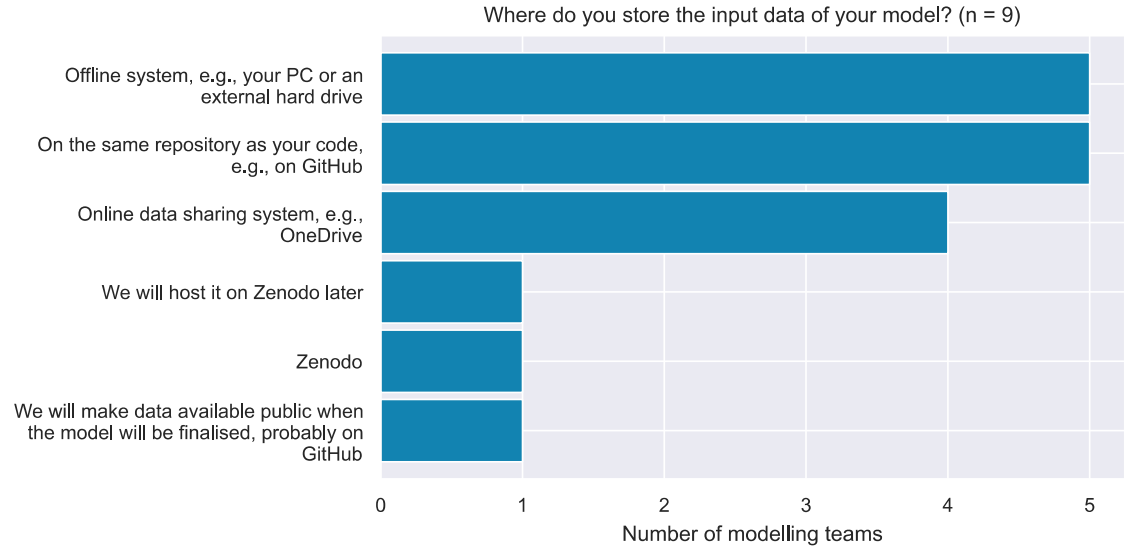


Figure 3. Current practices in data storage

In contrast to the previous practices, the remaining three practices of the model development category are not as popular among the respondents. Specifically, four modelling teams do not use any programming paradigms, design patterns, or coding standards in the development of their models (Figure 4). The rest report using at least one practice, mostly related to coding conventions and standards from their own modelling community. Additionally, most of the teams are not using an explicit lifecycle model such as agile development to guide

[25] GitHub suggests using the GitHub Large File Storage for files above 100 MB. For more details, see here: https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github.

modelling updates, with only three teams stating that they are using an iterative-like development (Figure 5). Refactoring methods also remain rather unused among the respondents, with only a handful suggesting that are using a test-driven development approach or a dedicated process based on GitHub Actions (Figure 6). These results are expected as most modelling teams do not have a member with SE background and these practices are not easy to apply without formal training and practice, despite them being the norm in the industry for many years now. This is also a finding for the general scientific software development community as many scientists that are developing software are using practices that relate to their scientific topic rather than with SE (Heaton & Carver, 2015). Despite the challenges of applying such practices, there are many potential benefits for IAM developers for adopting them. For instance, the use of a somewhat agile lifecycle model can correspond with the frequent calls for making IAMs more transparent, participatory, and responsive (Braunreiter et al., 2021; McGookin et al., 2024; Robertson, 2021).

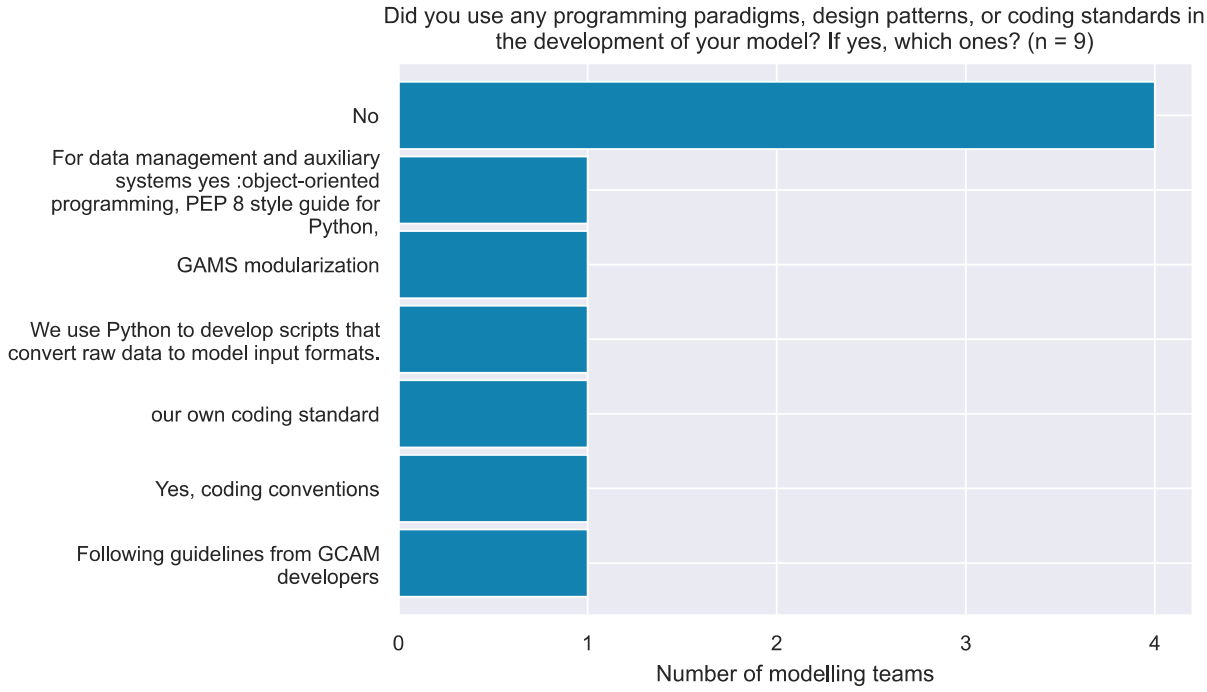


Figure 4. Current use of programming paradigms, design patterns, and coding standards

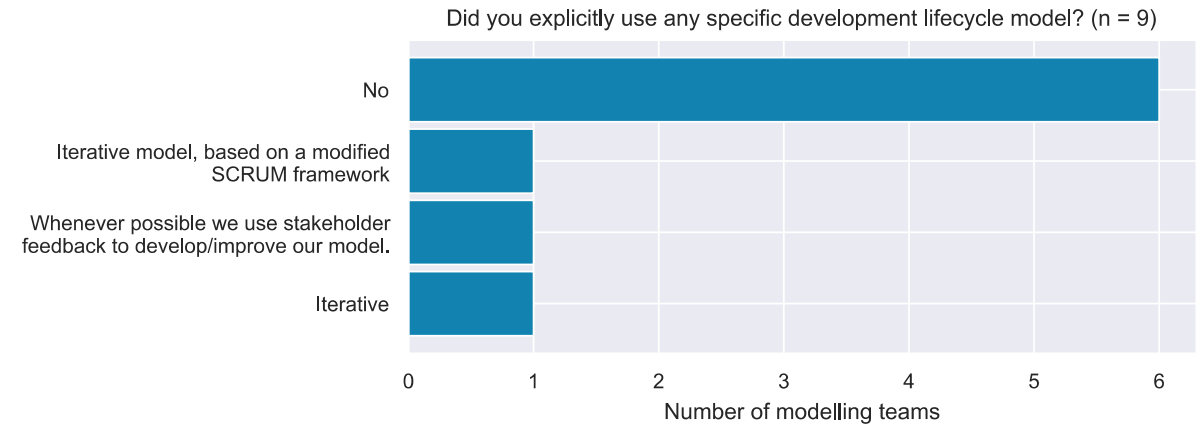


Figure 5. Current use of lifecycle models

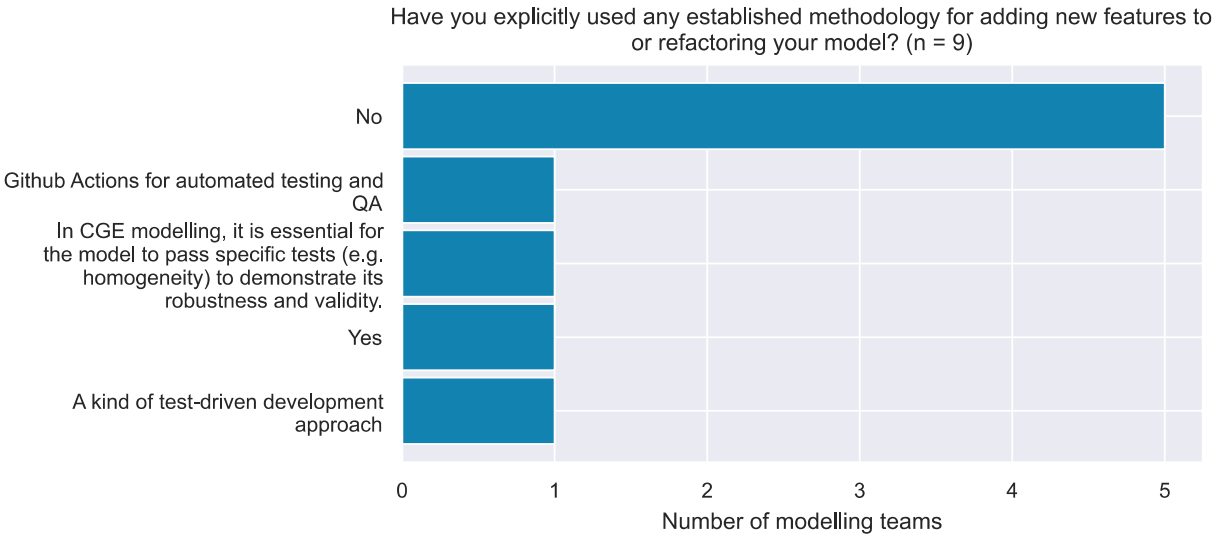


Figure 6. Current practices in code refactoring

Patterns on the relevance of these practices reflect the previous findings on their use: practices on requirements engineering and, especially, data processing are mentioned as relevant for most respondents while the rest are mentioned as less relevant, especially the ones related to lifecycle models (Figure 7). This may imply that the practices that the modellers are more comfortable with using are also the ones they find more relevant and vice versa.

Funded by the European Union

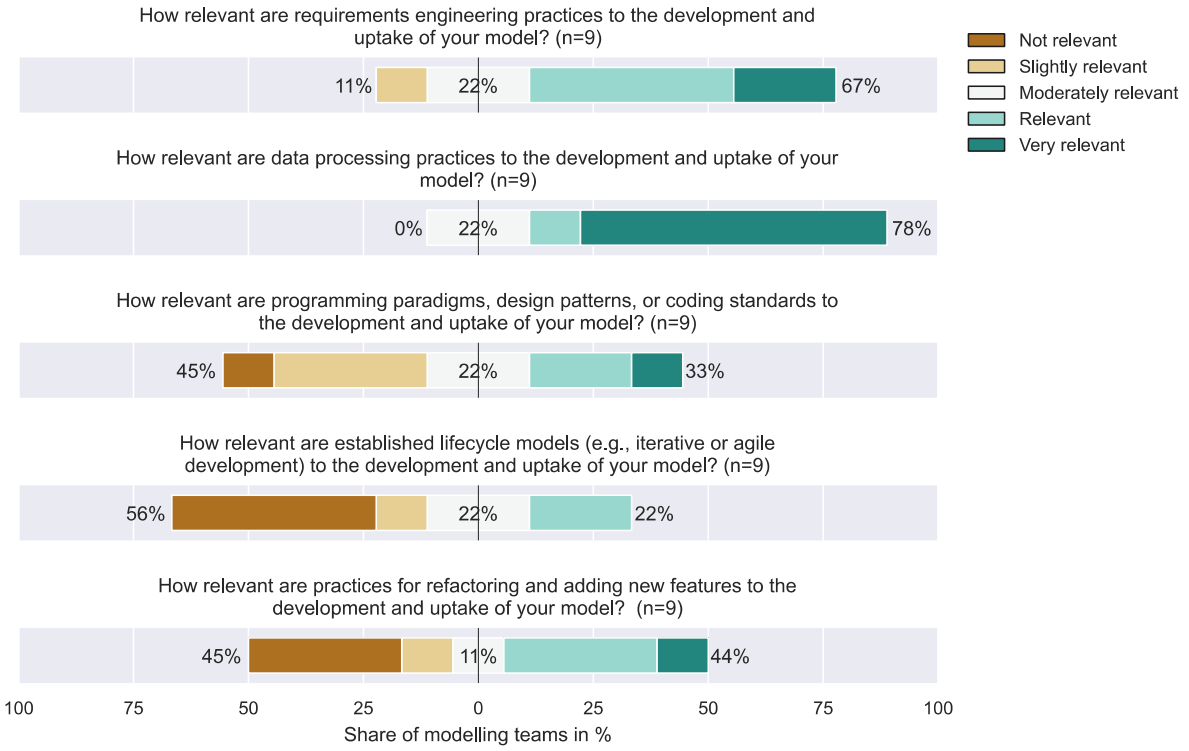


Figure 7. Relevance of model development practices

3.3 Documentation practices used

They were various suggestions for improving the conceptual documentation of models compared to the information that is usually included[26] (Table 3). Many of the suggestions focused on providing user guides and tutorials to help users run the model, while others recommended more details on the data and the equations used as well as providing validation test run for all new modelling features. While most of these suggestions support the fourth practice of the documentation category of the framework, which revolves around the development of tutorials, they reflect an intention to make the models more usable by a larger audience than the modelling teams that they develop them. Yet, three modelling teams do not have plans for developing a tutorial for their updated model during DIAMOND and only three suggesting that they will create one for the first version of their model as part of MS8 (Figure 8). This suggests that the upcoming work of Task 6.3 could focus on supporting more teams to develop a tutorial, facilitating future users and potentially driving the uptake of the improved models.

[26] Description of the purpose of the model, main components (algorithms, input data, and main outputs), key assumptions, uncertainties, pertinent features (technologies or policies modelled), as well as the temporal, spatial, and sectoral resolution of the model.

Table 3. Current practices in conceptual documentation

Model	Do you have any further suggestions on what to include in the conceptual documentation of our models apart from the aspects listed above[27]?
OMNIA	No
NEMESIS-World	List of model variables, equations, parameters in Annex
GCAM-Europe*	Validation test runs for new functions and data in the model More detailed instructions of data inputs and steps on how to run the model
GEMINI-E3 EU	No
openTEPES	No
CLEWS-EU	How to assemble the model, run it and visualise results.
ENGAGE	If the model is going to be open-source, we need to produce a user guide.
OPEN PROM	No

** The answers from the two GCAM-Europe teams have been merged into one.*

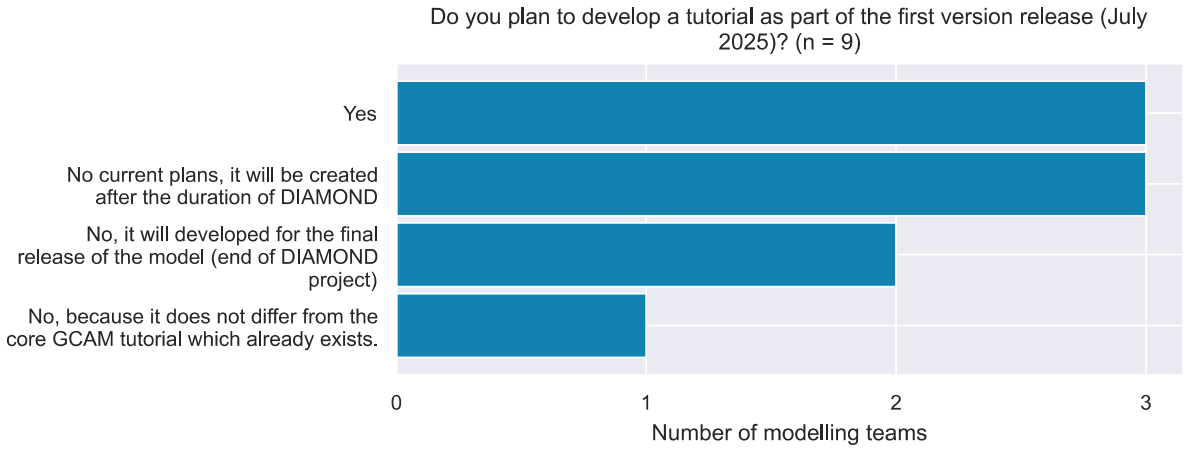


Figure 8. Current practices in tutorial development

In terms of code documentation, most modelling teams provide comments within the actual code (Figure 9), which is considered a good practice as such documentation can better adapt to code changes, compared to external documentation that is often difficult to maintain in line with the code (DeCarolis et al., 2012). Most teams also use an online documentation platform or wiki page to document their code as well as offline methods such as Word documents and README files. Similarly, most teams document their data with comments directly on the dataset (e.g., on the top of CSV files or in the first sheet of Excel files), while other teams provide documentation in README and GDX files, the latter being a kind of metadata file for models using the GAMS language for mathematical modelling. While providing documentation within the actual dataset can facilitate version control and keep track of data changes, it does not correspond to formal metadata standards such as Dublin Core. As the use of the IAMC template has helped standardize model results and facilitate their comparison, a standardized metadata solution or template for data inputs may also increase their findability and use by external users and enable data sharing between models, which would be useful for harmonisation purposes (Giarola et al., 2021).

[27] ibid

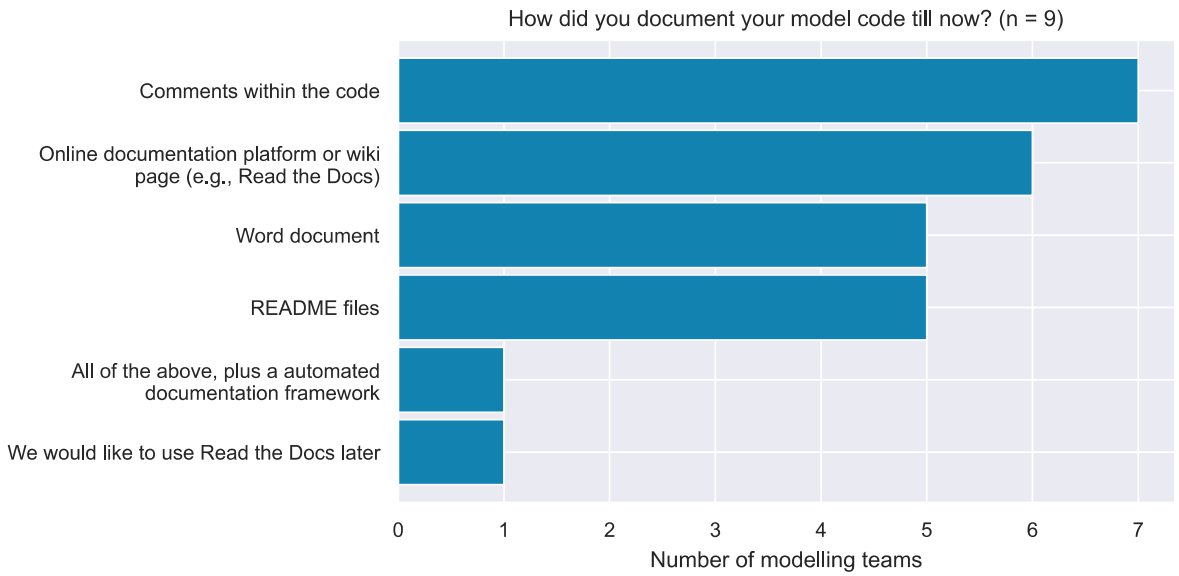


Figure 9. Current practices in code documentation

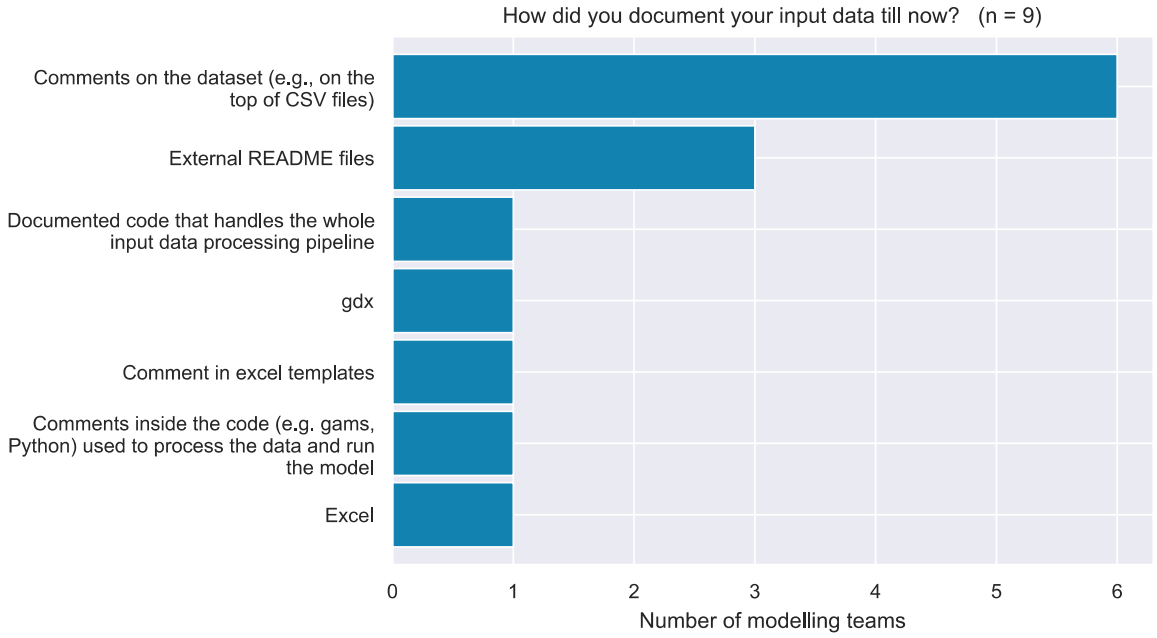


Figure 10. Current practices in data documentation

Most documentation methods appear as relevant or very relevant to most respondents (Figure 11). This is not surprisingly as model documentation was long assumed to be an important requirement for the credibility and transparency of IAMs (DeCarolis et al., 2012). The most relevant practice for the responding modellers revolves around code documentation. Still, it is unclear whether the current level of such documentation corresponds to coding conventions for readability such as PEP-8 in Python or is enough for external users to understand what is happening in the code. This is something that can be further explored as part of Task 6.3 in the code repositories of the models after MS8.

Funded by
the European Union

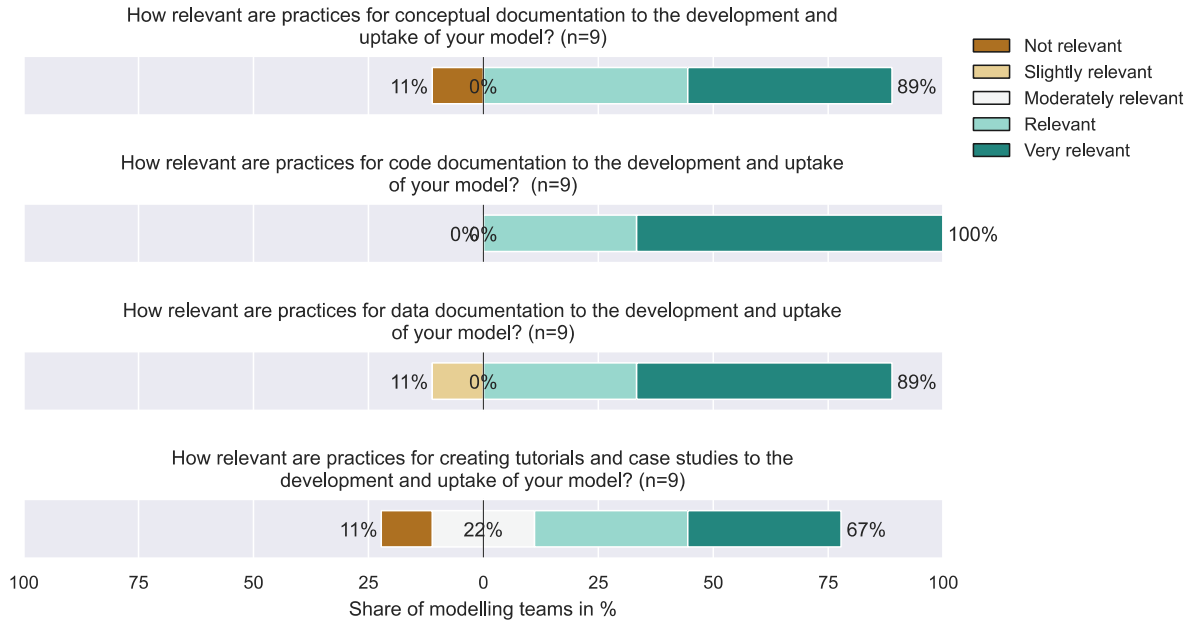


Figure 11. Relevance of documentation practices

3.4 Evaluation practices used

Expectedly, most responding teams have used a version of acceptability testing to test their model code with real-world data, while many have also performed end-to-end tests of the whole model (Figure 14). However, only two teams have written unit tests to evaluate whether the different functions of the model are performing as they should be. Similarly, only three teams have written regression tests when they add a new functionality to ensure that existing functionalities still work well. This lack of basic tests that are commonplace in the industry and a cornerstone of popular methods such as test-driven development can be potential explained by a comment from one of the teams that tests have been implemented in an ad-hoc manner, implying that the lack of SE training may hinder test implementation. Thus, providing training material that distil good testing practices and are adjusted to the needs of IAM development may help modelling teams to implement such methods more often. Subsequently, such material can enable automated testing, as only three teams currently use it while others may believe that it can be very complex to implement (Figure 13).

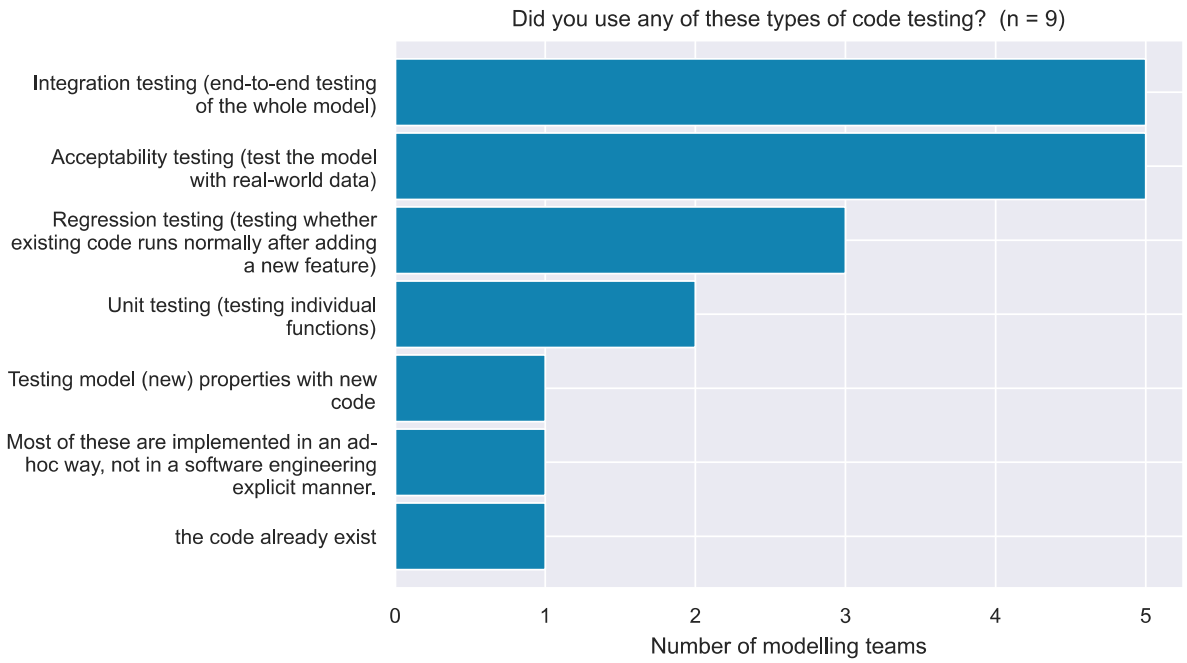


Figure 12. Current practices in code testing

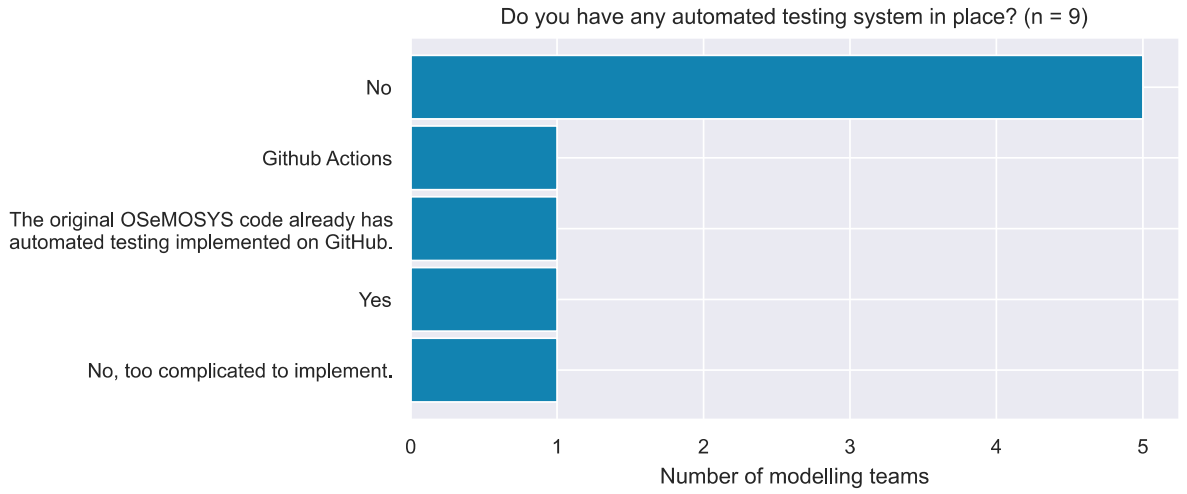


Figure 13. Current practices in automated testing systems

Most of the teams used a formal IAM evaluation method for their model, with most of them relying in sensitivity analyses and performing ex-post simulations to ensure that modelling results follow historical data (Figure 14). Four teams are aiming to use model intercomparisons to test their models, while four teams have examined near-term observations. Only two teams suggest that they are using or will be using the IPCC vetting process to check their results, despite its prevalence in the IPCC AR6 (Peters et al., 2023). Methods to automate this vetting process as part of DIAMOND and IAM COMPACT projects are already in development[28] and can potential help increase the number of teams that are using and passing vetting.

[28] https://github.com/ciceroOslo/iamcompact-validation-ui

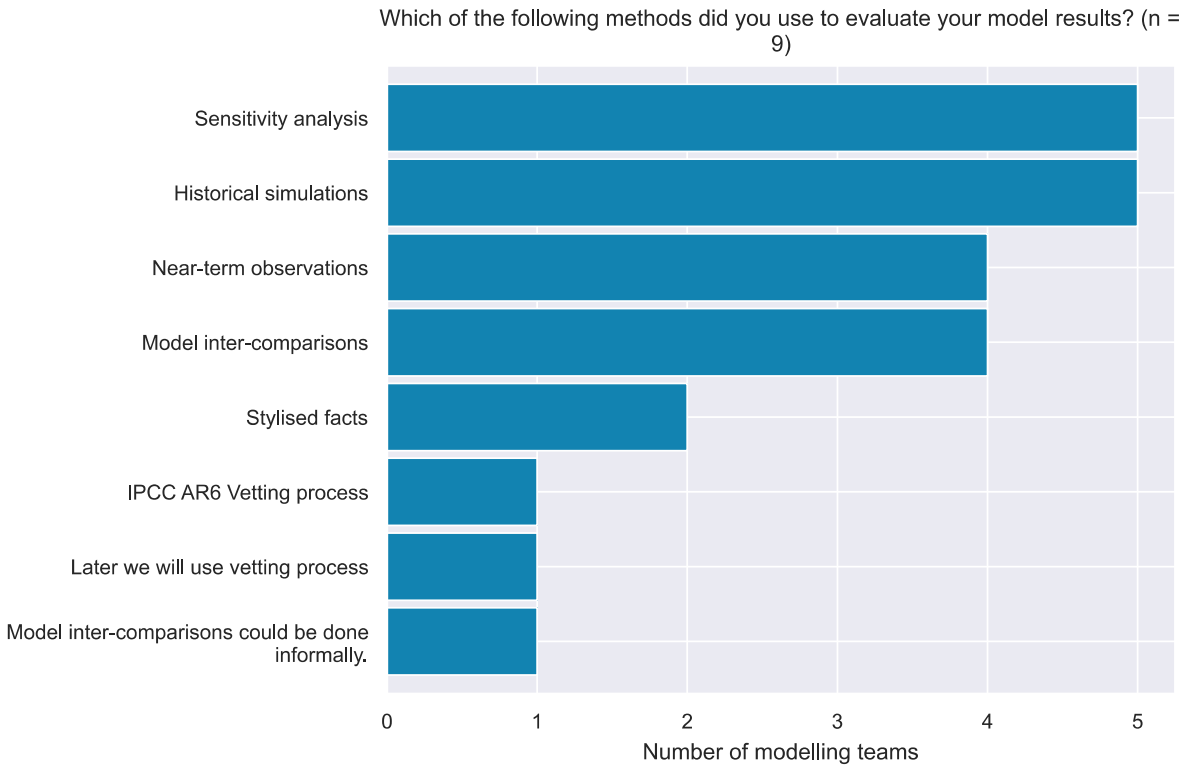


Figure 14. Current practices in model evaluation

All responding teams agree that formal evaluation practices are at least moderately relevant to their model, with many also agreeing that code testing is important (Figure 15). In contrast, many modellers are unsure about the relevance of automated testing methods, despite their prevalence in SE practice in the established paradigm of continuous integration. As discussed above, this can be supported by training material on testing methodologies and their automation, which can potentially facilitate their uptake.

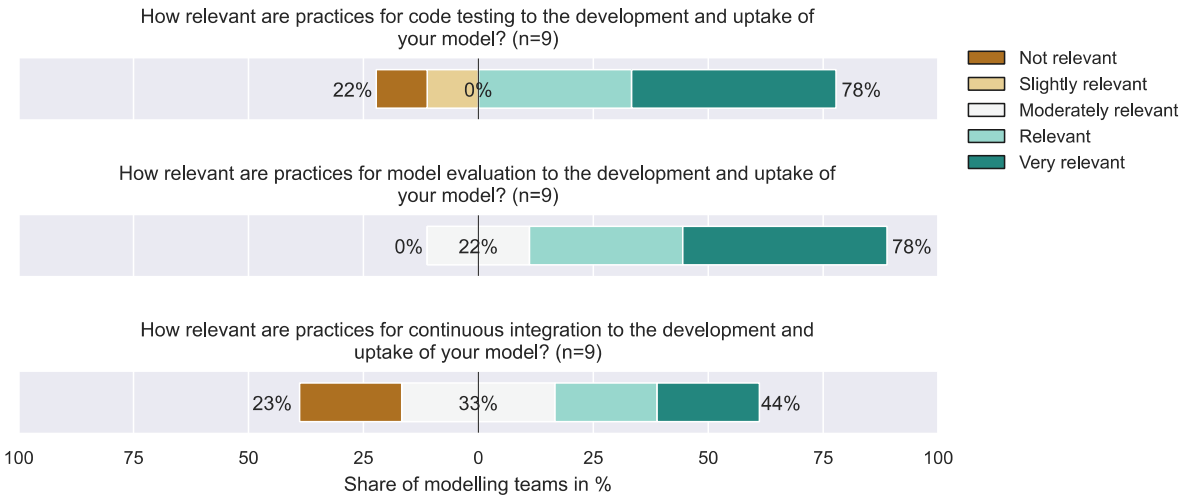


Figure 15. Relevance of evaluation practices

3.5 Versioning and collaboration practices used

As already shown in Section 3.2, most modelling teams use a version control system and plan to upload their model code in an online platform such as GitHub or GitLab (Figure 16). Only two WP3 modelling teams (Gemini-

E3 EU and NEMESIS-World) are not currently using such online platforms or version control in general and they may need support in enabling it in the future. While five responding teams are using the same online platforms for issue tracking, most teams use manual issue tracking systems, i.e., tracking development needs and bug fixes on paper, emails, Excel files etc. Thus, more information can be provided by Task 6.3 on the benefits of issue tracking when integrated with GitHub or other coding platforms. For instance, code changes can be related to issues and requests for bug fixes, helping model developers to keep track of their work and external users to understand how new features get implemented and how bugs are fixed in the code.

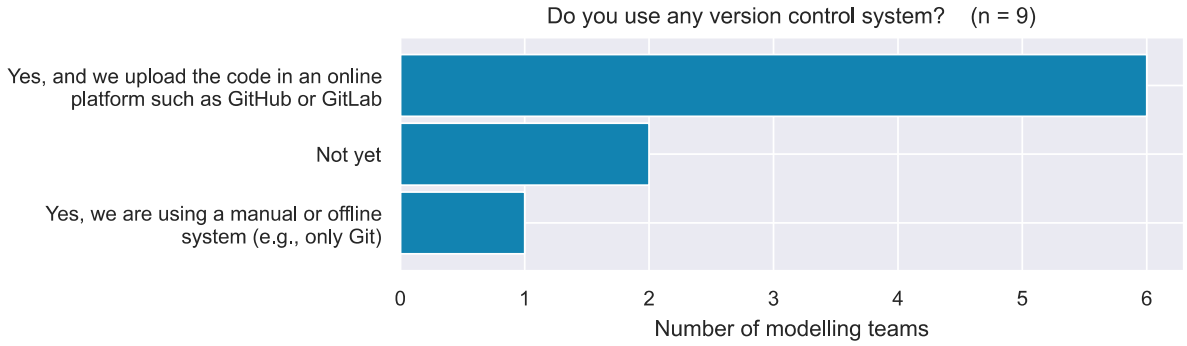


Figure 16. Current practices in version control

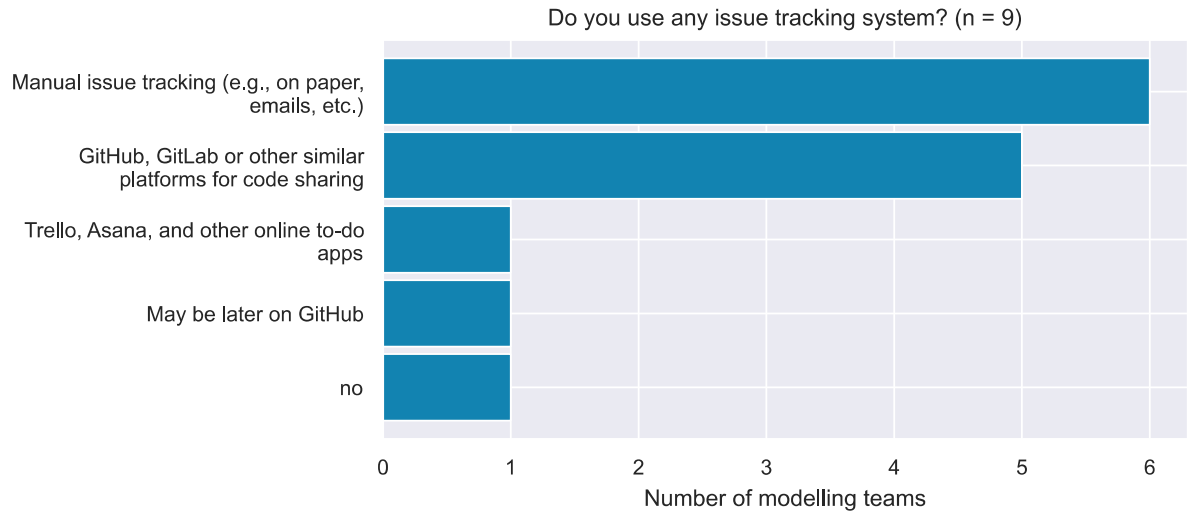


Figure 17. Current practices in issue tracking

In terms of open-source aspects, most models have specific licenses while others are still considering which one to use (Table 4). This may also indicate a point of attention for Task 6.3 to help model teams to choose an appropriate license. For instance, license types and their uses can be better explained, and current choices can be further examined, such as the use of CC-BY-NC-SA license for NEMESIS-World which better corresponds to data rather than model code. In terms of open data, most WP3 models use fully or predominantly open access datasets, including OMNIA, NEMESIS-World, GCAM-Europe, and CLEW-EU (Table 5). For other models, the use of proprietary datasets such as the most recent GTAP databases are not easy to bypass. Task 6.3 can also help here by providing other open access alternatives when possible. In addition, future work in Task 6.3 could focus on common licensing models for data, and potential pitfalls related to that. For instance, modelling teams may modify or merge existing datasets in a way that is not covered by the licenses of the original datasets. This may then require a new license to ensure that others can legally use their data.

Table 4. Current use of open-source licenses

Model	Which open-source license will you use for your model?
OMNIA	not yet finalised with the OMNIA team
NEMESIS-World	CC-BY-NC-SA
GCAM-Europe*	Educational community license 2.0
GEMINI-E3 EU	not yet fully determined
openTEPES	GNU AFFERO GPL3
CLEWS-EU	MIT
ENGAGE	ENGAGE is not an open-source model. We are not producing an open-source version of ENGAGE
OPEN-PROM	We will go with GNU AGPLv3 (https://choosealicense.com/licenses/agpl-3.0/) but we will add the "Commons Clause" to restrict commercial use: https://commonsclause.com/

** The answers from the two GCAM-Europe teams have been merged into one.*

Table 5. Current use of open data

Model	Is your input data open access? If yes, what is their license? If not, are there any alternative open access datasets that can be used?
OMNIA	yes
NEMESIS-World	Almost all, except one source without any alternative. Main sources: EXIOBASE (CC-BY-SA-NC), UN Energy Statistics Database processed by OMNIA team (free), OECD (free)
GCAM-Europe*	Yes, Eurostat: Creative Commons Attribution 4.0 International License
GEMINI-E3 EU	No, we use GTAP database
openTEPES	There are some case studies in GitHub
CLEWS-EU	It is open access data. Creative Commons Attribution-4.0 International license (Eurostat, JRC-IDEES, Faostat, EU-Reference scenario)
ENGAGE	No, the GTAP database requires a license. There are some alternative sources, but they require processing and adaptation.
OPEN-PROM	Most of them are not, but we don't distribute input data anyway. We show people how to generate them, though

** The answers from the two GCAM-Europe teams have been merged into one.*

Interestingly, all practices related to versioning and open access are deemed as relevant or very relevant by most of the teams in an almost uniform way (Figure 18). This aligns well with the fact that these practices can be often interconnected, for instance, GitHub provides both version control and issue tracking and is the most popular repository for open-source projects. Thus, providing information for integrating these practices can be another focus point for Task 6.3 in order to support modelling teams that are not currently using these practices and teams that already use them but in a disconnected manner.

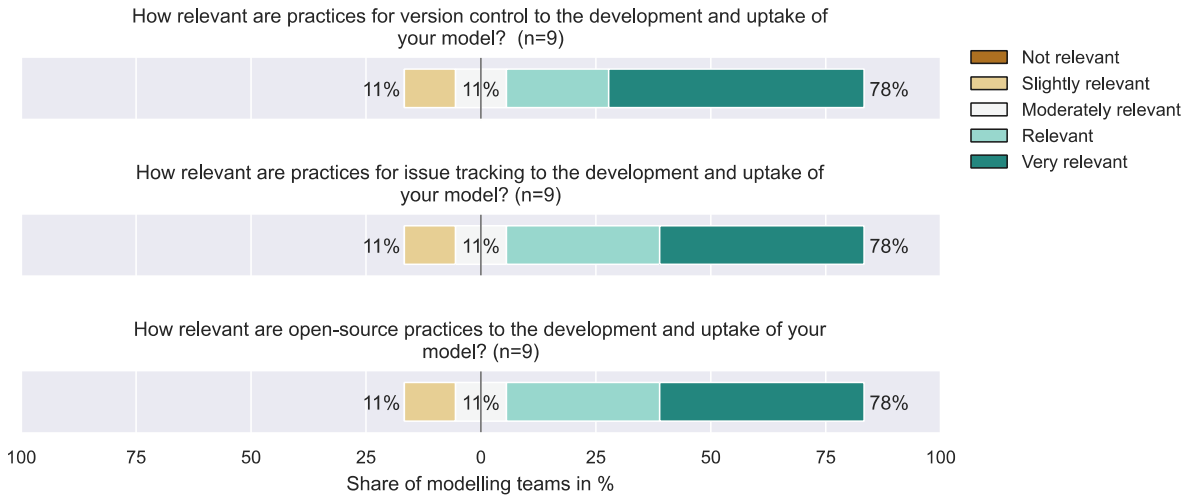


Figure 18. Relevance of versioning and collaboration practices

3.6 Deployment practices used

Most models involved in the survey have multiple dependencies, including mathematical modelling platforms such as GAMS, other libraries from the Python and R ecosystem, as well as linked/interrelated models or frameworks such as OSeMOSYS (Table 6). While for Python and R libraries there are structured methods to automatically manage and document dependencies (e.g., Poetry), only two modelling teams are reporting using them (Figure 19). In addition, three teams are using machine-readable formats such as requirements.txt and XML files while the other four teams are using less machine-readable formats such as README and Word documents. This can be a gap that Task 6.3 can address to facilitate model users and can be also combined with further work on tutorials. Also, Task 6.3 can provide some suggestions on managing dependencies that are not easily automated, such as GAMS installation. Docker could provide a useful application to manage all dependencies and help users install everything automatically in their machine or server. Since most modelling teams have not used Docker before (Figure 20), more information can be provided as part of Task 6.3 on how to make this application useful for IAM development and sharing.

Table 6. Current dependencies of the models

Model	What dependencies does your model currently have? If available, you can also provide a link to a requirements.txt file or a page that you document the dependencies.
OMNIA	GAMS and VEDA2.0 interface
NEMESIS-World	Iode software (https://github.com/plan-be/iode), and pyiode python package (https://readsthedocs.com/) than should be dependant from others Python packages, pymrio, pandas, numpy, MS Excel for data processing.
GCAM-Europe*	Dependency to tidyr 1.1.3, detailed in readme file. All requirements are detailed in the GCAM documentation to successfully run the model and to build the datasystem, the requirements are inherent to the gcamdata R package. Supply and demand curves, historical emissions and energy data, population and GDP and overall socioeconomic projections. See all sector inputs (supply, demand, land, ecomony) here https://jgcri.github.io/gcam-doc/index.html
GEMINI-E3 EU	no dependency

openTEPES	https://github.com/IIT-EnergySystemModels/openTEPES/blob/master/environment.yaml\ndependencies: altair=5.0.1\n - colorama=0.4.6\n - colour=0.1.5\n - gurobi=12.0.0\n - matplotlib=3.9.2\n - matplotlib-base=3.9.2\n - networkx=3.3\n - numpy=1.26.4\n - pandas>=2.2.2\n - pip=24.2\n - plotly=5.24.1\n - psutil=6.1.0\n - pip:\n - gurobipy==12.0.0\n - pyomo==6.8.1
CLEWS-EU	We use a YAML file to run OSeMOSYS models through Otoole. We will create a README in the model documentation that will detail the dependencies for running the CLEWS-EU model.
ENGAGE	It depends on the model application. Sometimes we link it with river systems model, material flow analysis models.
OPEN-PROM	GAMS (mandatory), R (optional)

* The answers from the two GCAM-Europe teams have been merged into one.

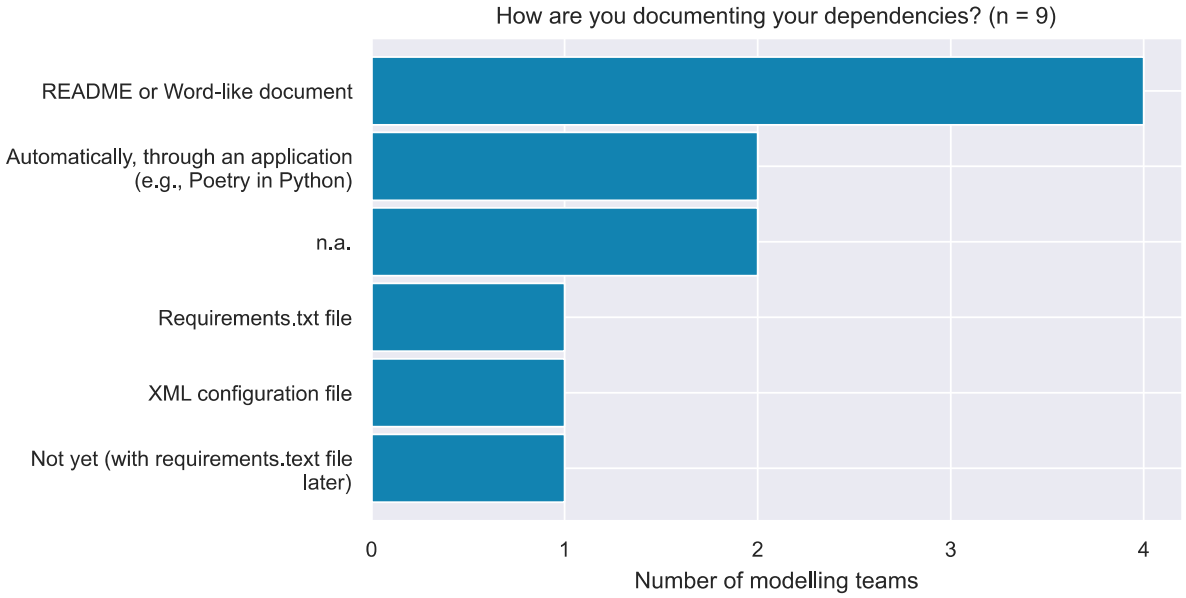


Figure 19. Current practices in dependencies documentation

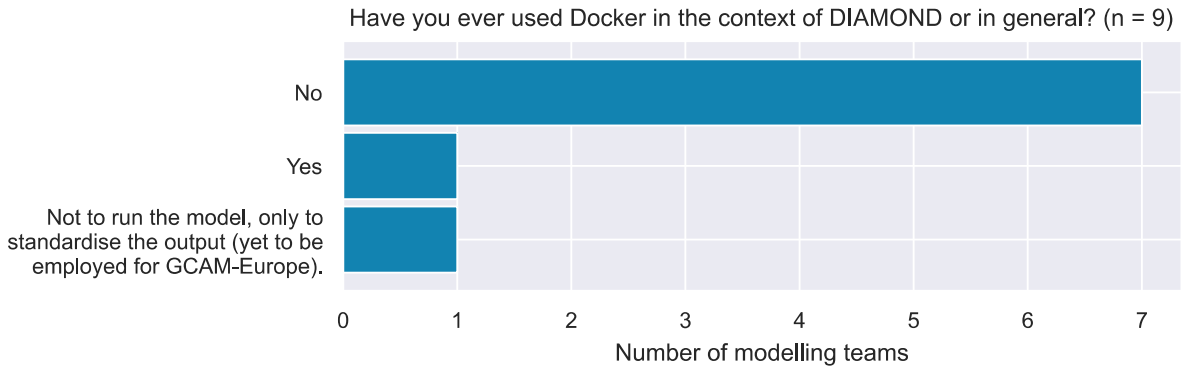


Figure 20. Current practices in Docker use

Most models have varying and relatively high system requirements (Table 7). The new IAM models OMNIA, CLEWs-EU and GCAM-Europe require a RAM of around 32 GB for most model runs, while the more detailed sectoral

models such as ENGAGE and OpenTEPES require almost three or four times this RAM. A potential activity as part of Task 6.3 could be to perform a sensitivity analysis of these system requirements in order to understand how they change along with the complexity of a model and allowing model users to select the right option for their application and system capacities. Also, a virtualisation application such as Docker would enable running the models in a high computing server, allowing users with lower system capacity to still use a model.

Table 7. Current system requirements of the models

Model	What are the (estimated) system requirements for running your model?
OMNIA	Hardware needed depends on the size and complexity of models, but here is a configuration suitable for typical TIMES models under Veda2.0:\n- Windows\n- CPU: Minimum 4 cores are recommended for STANDARD and ADVANCED licenses. 8 - 16 would be desirable for larger models\n- RAM: 4-8 GB is enough for Veda, but GAMS needs more RAM for larger models. 32 GB would accomodate most models\n- HDD: 500GB - 1TB free space for Veda and GAMS files
NEMESIS-World	No idea (laptop with relatively recent processors)
GCAM-Europe*	Any operating system. 16 GB memory recommended, computational power affects running time. Disk space for model itself ~18 GB. Every saved scenario occupies around 3 GB of additional space. GCAM requires significant computational resources. A GCAM model simulation will utilize over 8 GB of system RAM and storing the full results of the simulation will take around 3 GB of disk space per scenario.
GEMINI-E3 EU	Windows
openTEPES	Windows, MacOS, Linux\n128GB of memory for the European Case
CLEWS-EU	We have run only the aggregated and disaggregated sectoral models until now. These smaller models require 32 GB of RAM and an Intel Core i7-2.8 GHz processor. The model creates LP files during the matrix generation process, which needs a few GBs of hard disk memory. However, these files are temporary.
ENGAGE	It depends on the size of the model and application. For large models we use an in-house high-performance computer (e.g. 96 ram, 5.7 GHz CPU speed). For applications that require model linkages, iteration processes and machine learning, we use a supercomputer.
OPEN-PROM	>4GM RAM is the only system requirement

** The answers from the two GCAM-Europe teams have been merged into one.*

While many modelling teams agree that dependency management practices can be useful, they are more ambivalent or sceptical about practices related to system requirements management and continuous deployment such as the use of virtual machines or Docker containers (Figure 21). The latter is understandable as these practices have not been typically used in IAM development and require a significant time investment for setting up and running correctly. However, system requirements can be an important barrier for model users and can be better addressed in IAM development.

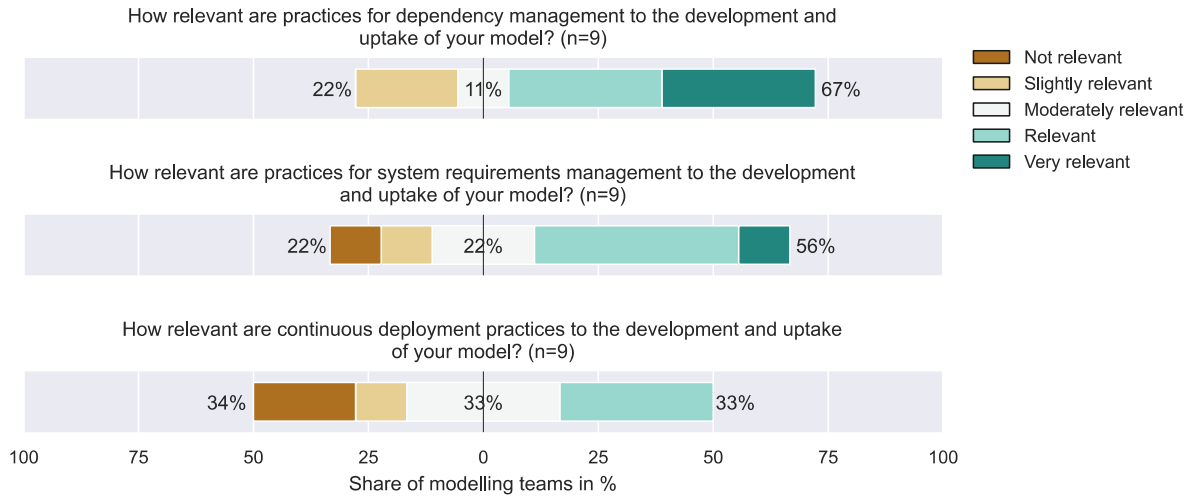


Figure 21. Relevance of deployment practices

3.7 Overall relevance of practices for DIAMOND models

Comparing among all 18 practices of the framework, it is clear that most practices related to documentation, versioning, and open access are considered as relevant or very relevant by the responding teams, along with practices related to data processing and model validation. Especially the increased scores for data-related practices such as data documentation and processing may indicate an increased attention of the responding modellers on his topic and thus a good focus point for future activities of Task 6.3. Interestingly, some of the most popular practices in current software engineering have been reported as less relevant for IAM development, such as lifecycle models, continuous integration, and continuous deployment. As reiterated above, the partners of Task 6.3 will aim to better communicate these benefits and identify easy ways to implement them in DIAMOND models, in close collaboration with WP3 and WP4 modellers.

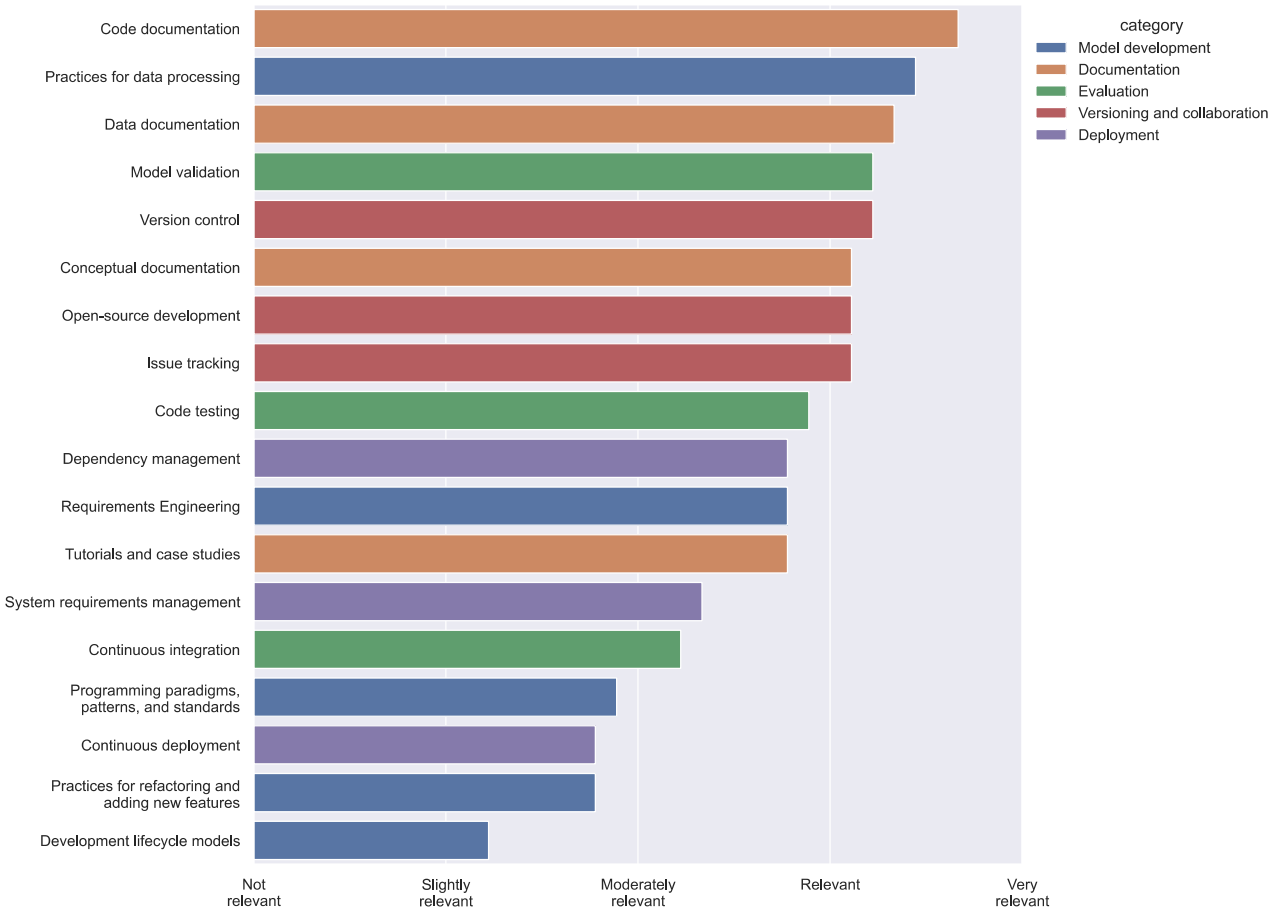


Figure 22. Mean relevance of all practices (N=9)

4 Conclusions and next steps

As shown throughout Section 3, the survey designed and rolled out in the context of the DIAMOND project provides several insights for improving model development in both DIAMOND and, potentially, in the wider modelling community. Most practices related to documentation, versioning, and open access were considered as relevant or very relevant by the responding teams, along with practices related to data processing and model validation. These results correspond to well-established requirements that have frequently been reiterated in the literature (DeCarolis et al., 2017; Pfenninger et al., 2018; C. Wilson et al., 2021), although their adoption by IAM teams has not always been widespread. For instance, almost all responding teams document input datasets using direct comments or external README files instead of structured metadata standards. In addition, many models use datasets that are not open access while alternative open datasets are not readily available. In terms of evaluation, most teams have performed sensitivity analysis on their models but do not yet have experience with the vetting process followed in the IPCC scenario processes of the AR6, despite its potential reuse in the AR7 cycle.

Other practices were deemed as less relevant for IAM development, including some of the most popular practices in SE, such as development lifecycle models, continuous integration, and continuous deployment. Most surveyed teams appear not to have used agile methods or any explicit lifecycle model to guide their development, established automated systems for testing or deploying, or tested individual functions and routines. Since most teams do not have members with SE backgrounds, it is understandable that they are hesitant to invest time and effort to implement such practices—especially when they may also lack the necessary know-how to do so effectively or when the positive impacts are not immediately clear to them. As such, practices perceived as of low relevance may still be extremely important to pursue (e.g., through standardised processes and training) to exploit untapped potential and improve the long-term sustainability and usability of the models. While these results are not necessarily representative of the broader modelling community, they indicate that current IAM development pipelines could be further improved based on good SE practices and that entry points may be needed for practices that modellers are still sceptical about.

As part of the next steps of Task 6.3, we will operationalise the framework by further clarifying the benefits of using modern SE methods for IAM development and identify easy ways to implement them, starting from the IAMs participating in the DIAMOND project. Specifically, we will provide concrete examples on how to apply these practices, using the CLEWS-EU model as a case study. This model was selected as its first version is available earlier than the other DIAMOND models, it is fully open-source, and its single-node version is relatively simpler and faster compared to the other models, making it ideal for capacity development exercises. In addition, we will aim to adapt and distribute the survey in the wider modelling community to uncover further evidence on the use of SE practices for modelling development and identify further practices that could be useful.

Building on the efforts of this study, a more systematic literature review can be performed to confirm and expand the practices included in the framework of Section 2 and provide more examples on how these practices are currently applied in the industry and in scientific software development. Along with the feedback from a follow-up survey with other modelling teams, an extensive literature review could also provide more details on the status quo of model development, and/or practical examples of solutions that would be suitable for use in the IAM community. All follow-up tutorials, checklists, and recommendations of Task 6.3 will be disseminated to the modelling community, to support the adoption of good development practices that can improve collaboration among, comprehensibility of, and trust in IAMs and relevant models.

5 References

Batini, C., & Scannapieco, M. (2016). *Data and Information Quality: Dimensions, Principles and Techniques*. Springer International Publishing. https://doi.org/10.1007/978-3-319-24106-7

Belete, G. F., Voinov, A., & Laniak, G. F. (2017). An overview of the model integration process: From pre-integration assessment to testing. *Environmental Modelling and Software*, *87*, 49–63. Scopus. https://doi.org/10.1016/j.envsoft.2016.10.013

Bond-Lamberty, B., Dorheim, K., Cui, R., Horowitz, R., Snyder, A., Calvin, K., Feng, L., Hoesly, R., Horing, J., Kyle, G. P., Link, R., Patel, P., Roney, C., Staniszewski, A., Turner, S., Chen, M., Feijoo, F., Hartin, C., Hejazi, M., … Clarke, L. (2019). gcamdata: An R Package for Preparation, Synthesis, and Tracking of Input Data for the GCAM Integrated Human-Earth Systems Model. *Journal of Open Research Software*, *7*(1). https://doi.org/10.5334/jors.232

Braunreiter, L., & Blumer, Y. B. (2018). Of sailors and divers: How researchers use energy scenarios. *Energy Research and Social Science*, *40*(July 2017), 118–126. https://doi.org/10.1016/j.erss.2017.12.003

Braunreiter, L., van Beek, L., Hajer, M., & van Vuuren, D. (2021). Transformative pathways – Using integrated assessment models more effectively to open up plausible and desirable low-carbon futures. *Energy Research and Social Science*, *80*(October). https://doi.org/10.1016/j.erss.2021.102220

Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, *139*, 32–50. https://doi.org/10.1016/j.jss.2018.01.036

David, O., Ascough, J. C., Lloyd, W., Green, T. R., Rojas, K. W., Leavesley, G. H., & Ahuja, L. R. (2013). A software engineering perspective on environmental modeling framework design: The Object Modeling System. *Environmental Modelling & Software*, *39*, 201–213. https://doi.org/10.1016/j.envsoft.2012.03.006

DeCarolis, J. F., Hannah Daly, Dodds, P., Keppo, I., Li, F., McDowall, W., Pye, S., Strachan, N., Trutnevyte, E., Usher, W., Winning, M., Yeh, S., & Zeyringer, M. (2017). Formalizing best practice for energy system optimization modelling. *Applied Energy*, *194*, 184–198. https://doi.org/10.1016/j.apenergy.2017.03.001

DeCarolis, J. F., Hunter, K., & Sreepathi, S. (2012). The case for repeatable analysis with energy economy optimization models. *Energy Economics*, *34*(6), 1845–1853. https://doi.org/10.1016/j.eneco.2012.07.004

Dekker, M., Hof, A., Du Pont, Y. R., Berg, N. V. D., Daioglou, V., Elzen, M. D., Heerden, R. V., Hooijschuur, E., Tagomori, I. S., Würschinger, C., & Vuuren, D. V. (2024). *Navigating the black box of fair national emissions targets*. https://doi.org/10.21203/rs.3.rs-5023350/v1

Demeter, M., Jele, A., & Major, Z. B. (2021). The International Development of Open Access Publishing: A Comparative Empirical Analysis Over Seven World Regions and Nine Academic Disciplines. *Publishing Research Quarterly*, *37*(3), 364–383. https://doi.org/10.1007/s12109-021-09814-9

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. https://doi.org/10.1016/j.jss.2012.02.033

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

Fisher-Vanden, K., & Weyant, J. (2020). The Evolution of Integrated Assessment: Developing the Next Generation of Use-Inspired Integrated Assessment Tools. *Annual Review of Resource Economics*, *12*(Volume 12, 2020), 471–487. https://doi.org/10.1146/annurev-resource-110119-030314

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176–189. https://doi.org/10.1016/j.jss.2015.06.063

Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code* (2nd edition). Addison-Wesley Professional.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education.

Giarola, S., Mittal, S., Vielle, M., Perdana, S., Campagnolo, L., Delpiazzo, E., Bui, H., Kraavi, A. A., Kolpakov, A., Sognnaes, I., Peters, G., Hawkes, A., Köberle, A. C., Grant, N., Gambhir, A., Nikas, A., Doukas, H., Moreno, J., & van de Ven, D. J. (2021). Challenges in the harmonisation of global integrated assessment models: A comprehensive methodology to reduce model response heterogeneity. *Science of the Total Environment*, *783*, 146861. https://doi.org/10.1016/j.scitotenv.2021.146861

Gidden, M., & Huppmann, D. (2019). pyam: A Python Package for the Analysis and Visualization of Models of the Interaction of Climate, Human, and Environmental Systems. *Journal of Open Source Software*, *4*(33), 1095. https://doi.org/10.21105/joss.01095

Hamilton, S. H., Fu, B., Guillaume, J. H. A., Badham, J., Elsawah, S., Gober, P., Hunt, R. J., Iwanaga, T., Jakeman, A. J., Ames, D. P., Curtis, A., Hill, M. C., Pierce, S. A., & Zare, F. (2019). A framework for characterising and evaluating the effectiveness of environmental modelling. *Environmental Modelling and Software*, *118*(August 2018), 83–98. https://doi.org/10.1016/j.envsoft.2019.04.008

Harmsen, M., Kriegler, E., Van Vuuren, D. P., Van Der Wijst, K.-I., Luderer, G., Cui, R., Dessens, O., Drouet, L., Emmerling, J., Morris, J. F., Fosse, F., Fragkiadakis, D., Fragkiadakis, K., Fragkos, P., Fricko, O., Fujimori, S., Gernaat, D., Guivarch, C., Iyer, G., … Zakeri, B. (2021). Integrated assessment model diagnostics: Key indicators and model evolution. *Environmental Research Letters*, *16*(5), 054046. https://doi.org/10.1088/1748-9326/abf964

Heaton, D., & Carver, J. C. (2015). Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology*, *67*, 207–219. https://doi.org/10.1016/j.infsof.2015.07.011

Hinkel, J. (2009). The PIAM approach to modular integrated assessment modelling. *Environmental Modelling & Software*, *24*(6), 739–748. https://doi.org/10.1016/j.envsoft.2008.11.005

Huppmann, D., Gidden, M., Fricko, O., Kolp, P., Orthofer, C., Pimmer, M., Kushin, N., Vinca, A., Mastrucci, A., Riahi, K., & Krey, V. (2019). The MESSAGEix Integrated Assessment Model and the *ix modeling platform* (ixmp): An open framework for integrated and cross-cutting analysis of energy, climate, the environment, and sustainable development. *Environmental Modelling & Software*, *112*, 143–156. https://doi.org/10.1016/j.envsoft.2018.11.012

Iwanaga, T., Rahman, J., Partington, D., Croke, B., & Jakeman, A. J. (2018). *Software Development Best Practices in Integrated Environmental Model Development*.

Jakeman, A. J., Letcher, R. A., & Norton, J. P. (2006). Ten iterative steps in development and evaluation of environmental models. *Environmental Modelling & Software*, *21*(5), 602–614. https://doi.org/10.1016/j.envsoft.2006.01.004

Keppo, I., Butnar, I., Bauer, N., Caspani, M., Edelenbosch, O., Emmerling, J., Fragkos, P., Guivarch, C., Harmsen, M., Lefevre, J., Le Gallic, T., Leimbach, M., Mcdowall, W., Mercure, J. F., Schaeffer, R., Trutnevyte, E., & Wagner, F. (2021). Exploring the possibility space: Taking stock of the diverse capabilities and gaps in integrated assessment models. *Environmental Research Letters*, *16*(5). https://doi.org/10.1088/1748-9326/abe5d8

Knapen, R., Verweij, P. J. F. M., & Janssen, S. (2010, January 1). Agilists and the Art of Integrated Assessment Tool

Development. *2010 International Congress on Environmental Modelling and Software*.

Krawczyk, F., & Braun, A. Ch. (2025). Models like heroes? Making Integrated Assessment Models (IAMs) ready for deep decarbonization and a socio-economic transformation. *Energy Research & Social Science*, *121*, 103959. https://doi.org/10.1016/j.erss.2025.103959

Lin, D., Crabtree, J., Dillo, I., Downs, R. R., Edmunds, R., Giaretta, D., De Giusti, M., L'Hours, H., Hugo, W., Jenkyns, R., Khodiyar, V., Martone, M. E., Mokrane, M., Navale, V., Petters, J., Sierman, B., Sokolova, D. V., Stockhause, M., & Westbrook, J. (2020). The TRUST Principles for digital repositories. *Scientific Data*, *7*(1), Article 1. https://doi.org/10.1038/s41597-020-0486-7

McGookin, C., Süsser, D., Xexakis, G., Trutnevyte, E., McDowall, W., Nikas, A., Koasidis, K., Few, S., Andersen, P. D., Demski, C., Fortes, P., Simoes, S. G., Bishop, C., Rogan, F., & Ó Gallachóir, B. (2024). Advancing participatory energy systems modelling. *Energy Strategy Reviews*, *52*, 101319. https://doi.org/10.1016/j.esr.2024.101319

Mikropoulos, E., Roelfsema, M., Chen, H.-H., Staffell, I., Oreggioni, G., Hdidouan, D., Thellufsen, J. Z., Chang, M. A., Fragkos, P., Giannousakis, A., Chatterjee, S., Ürge-Vorsatz, D., Pfenninger, S., Pickering, B., Victoria, M., Brown, T., & van Vuuren, D. P. (2025). Examining pathways for a climate neutral Europe by 2050; A model comparison analysis including integrated assessment models and energy system models. *Energy*, 134809. https://doi.org/10.1016/j.energy.2025.134809

Nikas, A., Gambhir, A., Trutnevyte, E., Koasidis, K., Lund, H., Thellufsen, J. Z., Mayer, D., Zachmann, G., Miguel, L. J., Ferreras-Alonso, N., Sognnaes, I., Peters, G. P., Colombo, E., Howells, M., Hawkes, A., van den Broek, M., Van de Ven, D. J., Gonzalez-Eguino, M., Flamos, A., & Doukas, H. (2021). Perspective of comprehensive and comprehensible multi-model energy and climate science in Europe. *Energy*, *215*, 119153. https://doi.org/10.1016/j.energy.2020.119153

Open Source Initiative. (2025). *The Open Source Definition*. https://opensource.org/osd

Peters, G. P., Al Khourdajie, A., Sognnaes, I., & Sanderson, B. M. (2023). AR6 scenarios database: An assessment of current practices and future recommendations. *Npj Climate Action*, *2*(1), Article 1. https://doi.org/10.1038/s44168-023-00050-9

Pfenninger, S., DeCarolis, J., Hirth, L., Quoilin, S., & Staffell, I. (2017). The importance of open data and software: Is energy research lagging behind? *Energy Policy*, *101*, 211–215. https://doi.org/10.1016/j.enpol.2016.11.046

Pfenninger, S., Hawkes, A., & Keirstead, J. (2014). Energy systems modeling for twenty-first century energy challenges. *Renewable and Sustainable Energy Reviews*, *33*, 74–86. Scopus. https://doi.org/10.1016/j.rser.2014.02.003

Pfenninger, S., Hirth, L., Schlecht, I., Schmid, E., Wiese, F., Brown, T., Davis, C., Gidden, M., Heinrichs, H., Heuberger, C., Hilpert, S., Krien, U., Matke, C., Nebel, A., Morrison, R., Müller, B., Pleßmann, G., Reeg, M., Richstein, J. C., … Wingenbach, C. (2018). Opening the black box of energy modelling: Strategies and lessons learned. *Energy Strategy Reviews*, *19*, 63–71. https://doi.org/10.1016/j.esr.2017.12.002

Plazas-Niño, F., Tan, N., Howells, M., Foster, V., & Quirós-Tortós, J. (2025). Uncovering the applications, developments, and future research directions of the open-source energy modelling system (OSeMOSYS): A systematic literature review. *Energy for Sustainable Development*, *85*, 101629. https://doi.org/10.1016/j.esd.2024.101629

Pohl, K. (2025). *Requirements Engineering*. Springer. https://link.springer.com/book/9783642125775

Robertson, S. (2021). Transparency, trust, and integrated assessment models: An ethical consideration for the Intergovernmental Panel on Climate Change. *WIREs Climate Change*, *12*(1), e679. https://doi.org/10.1002/wcc.679

Rodés-Bachs, C., Sampedro, J., Frilingou, N., Gardumi, F., & Lo Giudice, C. (2025). Open Science Practices in Integrated Assessment Models [version 1; peer review: Awaiting peer review]. *Open Research Europe*, *5*(12). https://doi.org/10.12688/openreseurope.18824.1

Rodés-Bachs, C., Sampedro, J., Horowitz, R., van de Ven, D.-J., Cui, R., Zhao, A., Zwerling, M., & Khan, Z. (2024). gcamreport: An R tool to process and standardize GCAM outputs. *Journal of Open Source Software*, *9*, 5975. https://doi.org/10.21105/joss.05975

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten Simple Rules for Reproducible Computational Research. *PLOS Computational Biology*, *9*(10), e1003285. https://doi.org/10.1371/journal.pcbi.1003285

Serrador, P., & Pinto, J. K. (2015). Does Agile work? - A quantitative analysis of agile project success. *International Journal of Project Management*, *33*(5), 1040–1051. Scopus. https://doi.org/10.1016/j.ijproman.2015.01.006

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

Siebers, P.-O., Lim, Z. E., Figueredo, G. P., & Hey, J. (2020). An Innovative Approach to Multi-Method Integrated Assessment Modelling of Global Climate Change. *Journal of Artificial Societies and Social Simulation*, *23*(1), 10.

Skea, J., Shukla, P., Al Khourdajie, A., & McCollum, D. (2021). Intergovernmental Panel on Climate Change: Transparency and integrated assessment modeling. *WIREs Climate Change*, *12*(5), e727. https://doi.org/10.1002/wcc.727

Soares, E., Sizilio, G., Santos, J., da Costa, D. A., & Kulesza, U. (2022). The effects of continuous integration on software development: A systematic literature review. *Empirical Software Engineering*, *27*(3), 78. https://doi.org/10.1007/s10664-021-10114-1

Stockhause, M., Huard, D., Khourdajie, A. A., Gutiérrez, J. M., Kawamiya, M., Klutse, N. A. B., Krey, V., Milward, D., Okem, A. E., Pirani, A., Sitz, L. E., Solman, S. A., Spinuso, A., & Xing, X. (2024). Implementing FAIR data principles in the IPCC seventh assessment cycle: Lessons learned and future prospects. *PLOS Climate*, *3*(12), e0000533. https://doi.org/10.1371/journal.pclm.0000533

Süsser, D., Gaschnig, H., Ceglarz, A., Stavrakas, V., Flamos, A., & Lilliestam, J. (2022). Better suited or just more complex? On the fit between user needs and modeller-driven improvements of energy system models. *Energy*, *239*, 121909. https://doi.org/10.1016/j.energy.2021.121909

Warren, R., de la Nava Santos, S., Arnell, N. W., Bane, M., Barker, T., Barton, C., Ford, R., Füssel, H.-M., Hankin, R. K. S., Klein, R., Linstead, C., Kohler, J., Mitchell, T. D., Osborn, T. J., Pan, H., Raper, S. C. B., Riley, G., Schellnhüber, H. J., Winne, S., & Anderson, D. (2008). Development and illustrative outputs of the Community Integrated Assessment System (CIAS), a multi-institutional modular integrated assessment approach for modelling climate change. *Environmental Modelling & Software*, *23*(5), 592–610. https://doi.org/10.1016/j.envsoft.2007.09.002

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., … Mons, B. (2016). Comment: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*, 1–9. https://doi.org/10.1038/sdata.2016.18

Wilson, C., Guivarch, C., Kriegler, E., van Ruijven, B., van Vuuren, D. P., Krey, V., Schwanitz, V. J., & Thompson, E. L. (2021). Evaluating process-based integrated assessment models of climate change mitigation. *Climatic Change*, *166*(1–2). https://doi.org/10.1007/s10584-021-03099-9

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, *13*(6), e1005510. https://doi.org/10.1371/journal.pcbi.1005510

6 Appendix

Table 8. Survey responses for the OMNIA model

Model	OMNIA
Organisation	E4SMA Srl
Developer in team	No
Requirements definition till now	Proposal call and Grant Agreement of DIAMOND; Ideas from the literature; Ideas from modelling teams beyond DIAMOND (e.g., GCAM community of practice); Experience in previous projects
Requirements engineering relevance	Slightly relevant
Data storage used	On the same repository as your code, e.g., on GitHub; Offline system, e.g., your PC or an external hard drive; Zenodo
Data processing practices	Use a version control system for monitoring changes in the data.; Process the data through scripts that are included in a version control system (e.g., GitHub).; Keep a backup of both raw and processed input data.
Data processing relevance	Relevant
Programming paradigms used	no
Programming paradigms relevance	Slightly relevant
Lifecycle models used	no
Lifecycle models relevance	Not relevant
Refactoring practices used	no
Refactoring practices relevance	Not relevant
Conceptual documentation apart the one in IAM PARIS	no
Conceptual documentation relevance	Relevant
Code documentation used	Online documentation platform or wiki page (e.g., Read the Docs); Word document
Code documentation relevance	Relevant
Data documentation used	Comments on the dataset (e.g., on the top of CSV files); Comment in excel templates
Data documentation relevance	Relevant
Tutorials developed	No, it will developed for the final release of the model (end of DIAMOND project)
Tutorials relevance	Relevant
Code testing used	the code already exist
Code testing relevance	Not relevant
Evaluation used	Model inter-comparisons; Sensitivity analysis
Evaluation relevance	Moderately relevant
Automatic testing	no
Continuous integration relevance	Not relevant
Version control use	Yes, and we upload the code in an online platform such as GitHub or GitLab
Version control relevance	Very relevant
Issue tracking use	Manual issue tracking (e.g., on paper, emails, etc.); GitHub, GitLab or other similar platforms for code sharing
Issue tracking relevance	Relevant
Open-source license	not yet finalised with the OMNIA team
Open data	yes
Open-source relevance	Moderately relevant
Dependencies	GAMS and VEDA2.0 interface
Dependencies documentation	n.a.
Dependencies relevance	Slightly relevant
System requirements	Hardware needed depends on the size and complexity of models, but here is a configuration suitable for typical TIMES models under Veda2.0:\n- Windows\n- CPU: Minimum 4 cores are recommended for STANDARD and ADVANCED licenses. 8 - 16 would be desirable for larger models\n- RAM: 4-8 GB is enough for Veda, but GAMS needs more RAM for larger models.

	32 GB would accomodate most models\n- HDD: 500GB - 1TB free space for Veda and GAMS files
System requirements relevance	Relevant
Docker use	no
Continuous deployment relevance	Moderately relevant
Other practices for the framework	nothing
Help needed for MS8	no

Table 9. Survey responses for the NEMESIS-World model

Model	NEMESIS-World
Organisation	SEURECO
Developer in team	No
Requirements definition till now	Proposal call and Grant Agreement of DIAMOND; Ideas from other modelling teams within DIAMOND; Ideas from the literature
Requirements engineering relevance	Relevant
Data storage used	Offline system, e.g., your PC or an external hard drive; We will make data available public when the model will be finalised, probably on GitHub
Data processing practices	Keep a backup of both raw and processed input data.; Use a version control system for monitoring changes in the data.
Data processing relevance	Very relevant
Programming paradigms used	No
Programming paradigms relevance	Slightly relevant
Lifecycle models used	No
Lifecycle models relevance	Slightly relevant
Refactoring practices used	A kind of test-driven development approach
Refactoring practices relevance	Relevant
Conceptual documentation apart the one in IAM PARIS	List of model variables, equations, parameters in Annex
Conceptual documentation relevance	Relevant
Code documentation used	Comments within the code; We would like to use Read the Docs later
Code documentation relevance	Very relevant
Data documentation used	NaN
Data documentation relevance	Slightly relevant
Tutorials developed	No current plans, it will be created after the duration of DIAMOND
Tutorials relevance	Not relevant
Code testing used	Unit testing (testing individual functions); Regression testing (testing whether existing code runs normally after adding a new feature); Integration testing (end-to-end testing of the whole model); Testing model (new) properties with new code
Code testing relevance	Very relevant
Evaluation used	Near-term observations; Sensitivity analysis; Later we will use vetting process
Evaluation relevance	Relevant
Automatic testing	No
Continuous integration relevance	Moderately relevant
Version control use	Yes, we are using a manual or offline system (e.g., only Git)
Version control relevance	Slightly relevant
Issue tracking use	Manual issue tracking (e.g., on paper, emails, etc.); May be later on GitHub
Issue tracking relevance	Moderately relevant
Open-source license	CC-BY-NC-SA
Open data	Almost all, except one source without any alternative. Main sources: EXIOBASE (CC-BY-SA-NC), UN Energy Statistics Database processed by OMNIA team (free), OECD (free)
Open-source relevance	Relevant
Dependencies	Iode software (https://github.com/plan-be/iode), and pyiode python package

	(https://readsthedocs.com/) than should be dependant from others Python packages, pymrio, pandas, numpy, MS Excel for data processing.
Dependencies documentation	Not yet (with requirements.text file later)
Dependencies relevance	Moderately relevant
System requirements	No idea (laptop with relatively recent processors)
System requirements relevance	Slightly relevant
Docker use	No
Continuous deployment relevance	Not relevant
Other practices for the framework	NaN
Help needed for MS8	After July, some "training sections"/discussions/exchange on how to properly use GitHub or ReadTheDocs could be really useful.

Table 10. Survey responses for the GCAM-Europe model

Model	GCAM-Europe	GCAM-Europe
Organisation	BC3	ICCS
Developer in team	No	Yes
Requirements definition till now	Proposal call and Grant Agreement of DIAMOND; Ideas from modelling teams beyond DIAMOND (e.g., GCAM community of practice)	Input from stakeholders through the engagement process in WP2; Proposal call and Grant Agreement of DIAMOND; Ideas from modelling teams beyond DIAMOND (e.g., GCAM community of practice); Ideas from other modelling teams within DIAMOND; Ideas from the literature
Requirements engineering relevance	Relevant	Relevant
Data storage used	On the same repository as your code, e.g., on GitHub	On the same repository as your code, e.g., on GitHub
Data processing practices	Use a version control system for monitoring changes in the data.; Process the data through scripts that are included in a version control system (e.g., GitHub).	Use a version control system for monitoring changes in the data.; Process the data through scripts that are included in a version control system (e.g., GitHub).; Keep a backup of both raw and processed input data.
Data processing relevance	Very relevant	Very relevant
Programming paradigms used	Following guidelines from GCAM developers	Yes, coding conventions
Programming paradigms relevance	Moderately relevant	Relevant
Lifecycle models used	No	Iterative
Lifecycle models relevance	Not relevant	Relevant
Refactoring practices used	No	Yes
Refactoring practices relevance	Not relevant	Relevant
Conceptual documentation apart the one in IAM PARIS	Validation test runs for new functions and data in the model	More detailed instructions of data inputs and steps on how to run the model
Conceptual documentation relevance	Relevant	Very relevant
Code documentation used	Comments within the code; Online documentation platform or wiki page (e.g., Read the Docs); README files	README files; Online documentation platform or wiki page (e.g., Read the Docs); Comments within the code
Code documentation relevance	Relevant	Very relevant
Data documentation used	Comments on the dataset (e.g., on the top of CSV files)	Comments on the dataset (e.g., on the top of CSV files)
Data documentation relevance	Relevant	Very relevant
Tutorials developed	No, because it does not differ from the core GCAM tutorial which already exists.	Yes
Tutorials relevance	Relevant	Very relevant
Code testing used	Acceptability testing (test the model with real-world data); Integration testing (end-to-end testing of the whole model)	Acceptability testing (test the model with real-world data)
Code testing relevance	Relevant	Very relevant
Evaluation used	Historical simulations; Near-term observations	Historical simulations

Evaluation relevance	Relevant	Very relevant
Automatic testing	No, too complicated to implement.	Yes
Continuous integration relevance	Moderately relevant	Relevant
Version control use	Yes, and we upload the code in an online platform such as GitHub or GitLab	Yes, and we upload the code in an online platform such as GitHub or GitLab
Version control relevance	Very relevant	Relevant
Issue tracking use	GitHub, GitLab or other similar platforms for code sharing; Trello, Asana, and other online to-do apps; Manual issue tracking (e.g., on paper, emails, etc.)	GitHub, GitLab or other similar platforms for code sharing
Issue tracking relevance	Relevant	Very relevant
Open-source license	Educational community license 2.0	ECL
Open data	Yes, Eurostat: Creative Commons Attribution 4.0 International License	Yes
Open-source relevance	Relevant	Very relevant
Dependencies	Dependency to tidyr 1.1.3, detailed in readme file. All requirements are detailed in the GCAM documentation to successfully run the model and to build the datasystem, the requirements are inherent to the gcamdata R package.	Supply and demand curves, historical emissions and energy data, population and GDP and overall socioeconomic projections. See all sector inputs (supply, demand, land, ecomony) here https://jgcri.github.io/gcam-doc/index.html
Dependencies documentation	Automatically, through an application (e.g., Poetry in Python); README or Word-like document	XML configuration file
Dependencies relevance	Relevant	Very relevant
System requirements	Any operating system. 16 GB memory recommended, computational power affects running time. Disk space for model itself ~18 GB. Every saved scenario occupies around 3 GB of additional space.	GCAM requires significant computational resources. A GCAM model simulation will utilize over 8 GB of system RAM and storing the full results of the simulation will take around 3 GB of disk space per scenario.
System requirements relevance	Moderately relevant	Relevant
Docker use	Not to run the model, only to standardise the output (yet to be employed for GCAM-Europe).	Yes
Continuous deployment relevance	Relevant	Relevant
Other practices for the framework	Very complete as it is.	Truly open access training, and data provision
Help needed for MS8	Standardisation of version numbering.	Transferring knowledge from the core dev team

Table 11. Survey responses for the GEMINI-E3 EU model

Model	**GEMINI-E3 EU**
Organisation	EPFL - LEURE
Developer in team	No
Requirements definition till now	Proposal call and Grant Agreement of DIAMOND; Ideas from other modelling teams within DIAMOND; Ideas from the literature
Requirements engineering relevance	Moderately relevant
Data storage used	Offline system, e.g., your PC or an external hard drive; Online data sharing system, e.g., OneDrive
Data processing practices	none
Data processing relevance	Moderately relevant
Programming paradigms used	no
Programming paradigms relevance	Slightly relevant
Lifecycle models used	no
Lifecycle models relevance	Not relevant
Refactoring practices used	no
Refactoring practices relevance	Moderately relevant
Conceptual documentation apart the one in IAM PARIS	no

Conceptual documentation relevance	Relevant
Code documentation used	Comments within the code; Word document
Code documentation relevance	Relevant
Data documentation used	Comments on the dataset (e.g., on the top of CSV files); gdx
Data documentation relevance	Relevant
Tutorials developed	No current plans, it will be created after the duration of DIAMOND
Tutorials relevance	Relevant
Code testing used	Acceptability testing (test the model with real-world data)
Code testing relevance	Relevant
Evaluation used	Model inter-comparisons; Sensitivity analysis; IPCC AR6 Vetting process
Evaluation relevance	Relevant
Automatic testing	no
Continuous integration relevance	Moderately relevant
Version control use	Not yet
Version control relevance	Moderately relevant
Issue tracking use	no
Issue tracking relevance	Slightly relevant
Open-source license	not yet fully determined
Open data	no we use GTAP database
Open-source relevance	Relevant
Dependencies	no dependency
Dependencies documentation	no
Dependencies relevance	Slightly relevant
System requirements	Windows
System requirements relevance	Not relevant
Docker use	no
Continuous deployment relevance	Moderately relevant
Other practices for the framework	nothing
Help needed for MS8	no

Table 12. Survey responses for the openTEPES model

Model	**openTEPES**
Organisation	Comillas Pontifical University
Developer in team	No
Requirements definition till now	Ideas from other modelling teams within DIAMOND
Requirements engineering relevance	Moderately relevant
Data storage used	On the same repository as your code, e.g., on GitHub
Data processing practices	Use a version control system for monitoring changes in the data.
Data processing relevance	Moderately relevant
Programming paradigms used	our own coding standard
Programming paradigms relevance	Moderately relevant
Lifecycle models used	No
Lifecycle models relevance	Not relevant
Refactoring practices used	No
Refactoring practices relevance	Not relevant
Conceptual documentation apart the one in IAM PARIS	No
Conceptual documentation relevance	Not relevant

Code documentation used	Comments within the code; README files; Online documentation platform or wiki page (e.g., Read the Docs)
Code documentation relevance	Very relevant
Data documentation used	External README files
Data documentation relevance	Very relevant
Tutorials developed	Yes
Tutorials relevance	Moderately relevant
Code testing used	Acceptability testing (test the model with real-world data)
Code testing relevance	Slightly relevant
Evaluation used	Stylised facts; Sensitivity analysis
Evaluation relevance	Moderately relevant
Automatic testing	No
Continuous integration relevance	Not relevant
Version control use	Yes, and we upload the code in an online platform such as GitHub or GitLab
Version control relevance	Very relevant
Issue tracking use	GitHub, GitLab or other similar platforms for code sharing
Issue tracking relevance	Very relevant
Open-source license	GNU AFFERO GPL3
Open data	There are some case studies in GitHub
Open-source relevance	Very relevant
Dependencies	https://github.com/IIT-EnergySystemModels/openTEPES/blob/master/environment.yaml\ndependencies:\n - altair=5.0.1\n - colorama=0.4.6\n - colour=0.1.5\n - gurobi=12.0.0\n - matplotlib=3.9.2\n - matplotlib-base=3.9.2\n - networkx=3.3\n - numpy=1.26.4\n - pandas>=2.2.2\n - pip=24.2\n - plotly=5.24.1\n - psutil=6.1.0\n - pip:\n - gurobipy==12.0.0\n - pyomo==6.8.1
Dependencies documentation	Requirements.txt file
Dependencies relevance	Very relevant
System requirements	Windows, MacOS, Linux\n128GB of memory for the European Case
System requirements relevance	Moderately relevant
Docker use	No
Continuous deployment relevance	Not relevant
Other practices for the framework	No
Help needed for MS8	No

Table 13. Survey responses for the CLEWS-EU model

Model	**CLEWS-EU**
Organisation	Imperial College London & The Cyprus Institute
Developer in team	No
Requirements definition till now	Proposal call and Grant Agreement of DIAMOND; Ideas from the literature; Ideas from modelling teams beyond DIAMOND (e.g., GCAM community of practice); Input from stakeholders through the engagement process in WP2
Requirements engineering relevance	Very relevant
Data storage used	Online data sharing system, e.g., OneDrive; On the same repository as your code, e.g., on GitHub; We will host it on Zenodo later
Data processing practices	Keep a backup of both raw and processed input data.; Process the data through scripts that are included in a version control system (e.g., GitHub).; version control only for the model and scripts, not input data
Data processing relevance	Very relevant
Programming paradigms used	We use Python to develop scripts that convert raw data to model input formats.
Programming paradigms relevance	Relevant
Lifecycle models used	No
Lifecycle models relevance	Moderately relevant

Refactoring practices used	No
Refactoring practices relevance	Slightly relevant
Conceptual documentation apart the one in IAM PARIS	How to assemble the model, run it and visualise results.
Conceptual documentation relevance	Very relevant
Code documentation used	Word document; Online documentation platform or wiki page (e.g., Read the Docs)
Code documentation relevance	Very relevant
Data documentation used	Excel; Comments on the dataset (e.g., on the top of CSV files)
Data documentation relevance	Very relevant
Tutorials developed	No, it will developed for the final release of the model (end of DIAMOND project)
Tutorials relevance	Very relevant
Code testing used	Integration testing (end-to-end testing of the whole model); Regression testing (testing whether existing code runs normally after adding a new feature); Most of these are implemented in an ad-hoc way, not in a software engineering explicit manner.
Code testing relevance	Relevant
Evaluation used	Historical simulations; Model inter-comparisons could be done informally.
Evaluation relevance	Very relevant
Automatic testing	The original OSeMOSYS code already has automated testing implemented on GitHub.
Continuous integration relevance	Relevant
Version control use	Yes, and we upload the code in an online platform such as GitHub or GitLab
Version control relevance	Relevant
Issue tracking use	Manual issue tracking (e.g., on paper, emails, etc.)
Issue tracking relevance	Relevant
Open-source license	MIT
Open data	It is open access data. Creative Commons Attribution-4.0 International license (Eurostat, JRC-IDEES, Faostat, EU-Reference scenario)
Open-source relevance	Very relevant
Dependencies	We use a YAML file to run OSeMOSYS models through Otoole. We will create a README in the model documentation that will detail the dependencies for running the CLEWS-EU model.
Dependencies documentation	README or Word-like document
Dependencies relevance	Very relevant
System requirements	We have run only the aggregated and disaggregated sectoral models until now. These smaller models require 32 GB of RAM and an Intel Core i7-2.8 GHz processor. The model creates LP files during the matrix generation process, which needs a few GBs of hard disk memory. However, these files are temporary.
System requirements relevance	Very relevant
Docker use	No
Continuous deployment relevance	Slightly relevant
Other practices for the framework	NaN
Help needed for MS8	We will connect with the relevant WPs in the coming months if and when required.

Table 14. Survey responses for the ENGAGE model

Model	ENGAGE
Organisation	UCL
Developer in team	No
Requirements definition till now	Input from stakeholders through the engagement process in WP2
Requirements engineering relevance	Relevant
Data storage used	Offline system, e.g., your PC or an external hard drive; Online data sharing system, e.g., OneDrive
Data processing practices	Keep a backup of both raw and processed input data.; Use a version control system for monitoring changes in the data.
Data processing relevance	Very relevant

Programming paradigms used	No
Programming paradigms relevance	Not relevant
Lifecycle models used	Whenever possible we use stakeholder feedback to develop/improve our model.
Lifecycle models relevance	Moderately relevant
Refactoring practices used	In CGE modelling, it is essential for the model to pass specific tests (e.g. homogeneity) to demonstrate its robustness and validity.
Refactoring practices relevance	Very relevant
Conceptual documentation apart the one in IAM PARIS	If the model is going to be open-source, we need to produce a user guide.
Conceptual documentation relevance	Very relevant
Code documentation used	Comments within the code; README files; Word document
Code documentation relevance	Very relevant
Data documentation used	Comments inside the code (e.g. gams, Python) used to process the data and run the model; External README files
Data documentation relevance	Very relevant
Tutorials developed	No current plans, it will be created after the duration of DIAMOND
Tutorials relevance	Moderately relevant
Code testing used	Regression testing (testing whether existing code runs normally after adding a new feature); Integration testing (end-to-end testing of the whole model); Unit testing (testing individual functions); Acceptability testing (test the model with real-world data)
Code testing relevance	Very relevant
Evaluation used	Historical simulations; Near-term observations; Stylised facts; Model inter-comparisons; Sensitivity analysis
Evaluation relevance	Very relevant
Automatic testing	No
Continuous integration relevance	Very relevant
Version control use	Not yet
Version control relevance	Very relevant
Issue tracking use	Manual issue tracking (e.g., on paper, emails, etc.)
Issue tracking relevance	Very relevant
Open-source license	ENGAGE is not an open-source model. We are not producing an open-source version of ENGAGE
Open data	No, the GTAP database requires a license. There are some alternative sources, but they require processing and adaptation.
Open-source relevance	Slightly relevant
Dependencies	It depends on the model application. Sometimes we link it with river systems model, material flow analysis models.
Dependencies documentation	Automatically, through an application (e.g., Poetry in Python); README or Word-like document
Dependencies relevance	Relevant
System requirements	It depends on the size of the model and application. For large models we use an in-house high performance computer (e.g. 96 ram, 5.7 GHz CPU speed). For applications that require model linkages, iteration processes and machine learning, we use a super computer.
System requirements relevance	Relevant
Docker use	No
Continuous deployment relevance	Moderately relevant
Other practices for the framework	I think knowing the number of people working on the model helps to understand the context of some answers. In the case of ENGAGE, we are normally 3 researchers working on different different versions and applications of the model.
Help needed for MS8	No.

Table 15. Survey responses for the OPEN-PROM model

Model	**OPEN-PROM**
Organisation	ICCS/E3 Modelling

Developer in team	Yes
Requirements definition till now	Proposal call and Grant Agreement of DIAMOND; Input from stakeholders through the engagement process in WP2; Ideas from the literature; Ideas from other modelling teams within DIAMOND
Requirements engineering relevance	Very relevant
Data storage used	Online data sharing system, e.g., OneDrive; Offline system, e.g., your PC or an external hard drive
Data processing practices	Use a version control system for monitoring changes in the data.; Keep a backup of both raw and processed input data.; Process the data through scripts that are included in a version control system (e.g., GitHub).
Data processing relevance	Very relevant
Programming paradigms used	GAMS modularization; For data management and auxiliary systems yes :object-oriented programming, PEP 8 style guide for Python,
Programming paradigms relevance	Very relevant
Lifecycle models used	Iterative model, based on a modified SCRUM framework
Lifecycle models relevance	Relevant
Refactoring practices used	Github Actions for automated testing and QA
Refactoring practices relevance	Relevant
Conceptual documentation apart the one in IAM PARIS	No
Conceptual documentation relevance	Very relevant
Code documentation used	Comments within the code; README files; Word document; Online documentation platform or wiki page (e.g., Read the Docs); All of the above, plus a automated documentation framework
Code documentation relevance	Very relevant
Data documentation used	Comments on the dataset (e.g., on the top of CSV files); External README files; Documented code that handles the whole input data processing pipeline
Data documentation relevance	Very relevant
Tutorials developed	Yes
Tutorials relevance	Very relevant
Code testing used	Integration testing (end-to-end testing of the whole model)
Code testing relevance	Very relevant
Evaluation used	Historical simulations; Near-term observations; Model inter-comparisons
Evaluation relevance	Very relevant
Automatic testing	Github Actions
Continuous integration relevance	Very relevant
Version control use	Yes, and we upload the code in an online platform such as GitHub or GitLab
Version control relevance	Very relevant
Issue tracking use	GitHub, GitLab or other similar platforms for code sharing; Manual issue tracking (e.g., on paper, emails, etc.)
Issue tracking relevance	Very relevant
Open-source license	We will go with GNU AGPLv3 (https://choosealicense.com/licenses/agpl-3.0/) but we will add the "Commons Clause" to restrict commercial use: https://commonsclause.com/
Open data	Most of them are not, but we don't distribute input data anyway. We show people how to generate them, though
Open-source relevance	Very relevant
Dependencies	GAMS (mandatory), R (optional)
Dependencies documentation	README or Word-like document
Dependencies relevance	Relevant
System requirements	>4GM RAM is the only system requirement
System requirements relevance	Relevant
Docker use	no
Continuous deployment relevance	Relevant

Other practices for the framework	The list is complete
Help needed for MS8	No, thanks