Weather Generator

D3.1 -- WeatherGenerator software development roadmap

Due date of deliverable	31 July 2025
Submission date	
File Name	WeatherGenerator-D3.1-WP3-V1.0.docx
Work Package /Task	WP3 / WeatherGenerator software development roadmap
Organisation Responsible of Deliverable	FZJ (Reviewers: MPG, ECMWF)
Author name(s)	Martin Schultz, Timothee Hunter, Christian Lessig, Ankit Patnala, Michael Langguth, Vitus Benson, Sebastian Hoffmann, Florentine Weber
Revision number	1.0
Status	final
Dissemination Level	Public

Funded by the European Union

The WeatherGenerator project (grant agreement No 101187947) is funded by the European Union.
Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Commission. Neither the European Union nor the granting authority can be held responsible for them.

1 Executive Summary

The development of a new modelling system for weather (and climate) applications requires a stringent, traceable, and reproducible approach to software development. In the WeatherGenerator project, this also has to take into account that most of the contributing code developers are scientists rather than professional software engineers and that the code will have to meet very high scientific standards. This implies that the practices for code development that have been adopted in the WeatherGenerator project need to combine scientific goals like flexibility and ease of use with technical objectives like reproducibility, performance, portability, and a clean code structure. However, software development is not just a technical affair, but it also hinges strongly on the community of people who are developing the code. Therefore, this deliverable document summarizes the procedures that have been adopted in the WeatherGenerator project from a larger perspective. It includes technical descriptions, for example of the repository structure and the handling of GitHub issues, as well as procedural information, e.g., how often the developers meet and how new people are brought into the development process. This “roadmap” document presents a snapshot of WeatherGenerator development practices five months into the project. While many things have been consolidated in the meantime, other aspects, such as the release policy and procedures, will necessarily have to develop further over time. Up-to-date information about the topics described in this document will be made available in the central GitHub repository and its associated Wiki.

Table of Contents

2 Introduction

2.1 Background

Climate change brings huge risks for the wellbeing and prosperity of society in Europe and world-wide. Earth system models that provide a numerical representation of the various components of the Earth system with atmosphere, ocean, land surface, land ice, sea ice, lakes and atmospheric chemistry are currently the best available tools to understand and prepare for climate change and the associated weather extremes.

The WeatherGenerator project will build the world's best generative Foundation Model of the Earth system – that will serve as a new Digital Twin for Destination Earth[1]. The WeatherGenerator will be based on representation learning and create a general and versatile tool that models the dynamics of the Earth system based on a large variety of Earth system data. At the same time, it will integrate observations and simulations at a previously unseen level and scale.

This project brings together Europe's leading scientific groups and research institutes, Small and Medium-sized Enterprises (SMEs), and industry partners in the area of Earth system modelling, high-performance computing (HPC) and machine learning to build the WeatherGenerator as a new Digital Twin of DestinE. Once trained, the WeatherGenerator will be applied for selected high-impact applications in the energy, food, water and health sectors.

The WeatherGenerator will lead to key innovations in weather and climate science and machine learning to enable Europe to establish and defend leadership with respect to machine learning based Earth system modelling. The WeatherGenerator will define a new state-of-the-art in both machine learning and weather and climate sciences. Through its vastly improved efficiency and flexibility compared to current Earth system models, the WeatherGenerator will create new opportunities for fast DestinE services that allow testing of many different management options and can include new levels of interactivity for a large user base including, for example, city planners, regional and national authorities, architects, and engineering companies.

Specifically, WP3 has as its main objective the development of the prototype WeatherGenerator model. This requires substantial developments of machine learning methods and methodology to enable the Earth system science community to leverage the full potential of artificial intelligence and develop a machine learned Foundation Model for weather and climate applications. At the end of this work package, pre-trained versions of WeGen-Atmo-S and WeGen-Land will be made available, utilising a subset of the available input data, including a mixture of reanalyses, simulation output and observations.

2.2 Scope of this deliverable

2.2.1 Objectives of this deliverable

This deliverable contains the WeatherGenerator software development roadmap and is tied to WP3, task 3.1.

This deliverable summarizes the concepts that have been developed and the procedures that have been put in place to organize the software development around the WeatherGenerator core model, enable efficient collaboration among the contributing partners, and ensure that functional versions of the model are available when needed.

This report collects and complements information that is given on the collaborative platforms that are used for the WeatherGenerator development, i.,e., foremost:

[1] https://destination-earth.eu/: Also abbreviated as DestinE.

- The GitHub repository at https://GitHub.com/ecmwf/WeatherGenerator
- The Wiki at https://gitlab.jsc.fz-juelich.de/esde/WeatherGenerator-private/-/wikis/home (restricted access)
- The collection of design documents (accessible to project partners only on shared OpenOffice repository)
- The WeatherGenerator-Land GitHub repository at https://GitHub.com/EarthyScience/WeatherGenerator-Land.git

For up-to-date information, the resources listed above should be consulted. This document will not be maintained as a living document, instead, an updated version shall be produced in project phase 2 in the winter of 2026/27 (D4.1).

2.2.2 Work performed in this deliverable

In this deliverable, the work as planned in the Description of Action (DoA, WP3 D3.1) was performed.

The WeatherGenerator software development requires a structured approach to enable efficient collaboration among the contributing partners and to ensure that functional versions of the model are available when needed, for example for the downstream applications. In task 3.1, we have set-up the required software repository, defined how to manage issues and versions, define releases, and periodically update development roadmaps for the core WeatherGenerator model. This enables other project activities and in particular the downstream applications from Theme 3 to take this into consideration in their planning. In this sense, task 3.1 builds the foundation for all core model development in Theme 2.

The general design goal of the WeatherGenerator core model – beyond its scientific quality and versatility -- is that any developer can take the latest develop branch of the code, launch a training run on their HPC environment and evaluate the output without modification of the code and at any time. At least on those HPC systems that are officially adopted by the project (i.e., at present, JUWELS Booster and JURECA at JSC, Leonardo at CINECA, Levante at DKRZ, Marenostrum at BSC, and Alps at CSCS), there should be no manual fiddling to train a model.

Based on an initial conceptual model prototype that was made available on GitHub by Christian Lessig (ECMWF) at the start of the project, a developer team was formed with scientists and software engineers from the partner institutions involved. A design document with suggestions for coding practices and the GitHub repository structure was drafted and discussed during a dedicated roadmap meeting. Design documents, GitHub issues, and pull requests were identified as key tools for the collaboration, and weekly developer meetings were set in place to discuss the state of the development, open issues, and other matters. Once per month, a wider perspective on the software status involving both technical and scientific aspects is discussed in so-called roadmap meetings. Furthermore, a series of hackathon events was organized to facilitate code adoption and provide opportunities for in-depth discussions and code sprints. A release strategy has been defined only loosely so far, awaiting finalisation after the first prototype version shall be released in month 7 (August 2025).

The coding and collaboration practices adopted for the WeatherGenerator development follow SCRUM principles although deadlines are less stringent and some adaptations were made to better accommodate the rhythm and specialities of scientific developments.

The work associated with this deliverable also includes the development of a Continuous Integration Continuous Deployment (CI/CD) workflow for the WeatherGenerator, work on which has started but is not completed yet.

The management of (larger) input datasets is of course also relevant to the core model development. However, this is part of Theme 1, WP1, and described in deliverable D1.1. On the other hand, the storage of model output and, more generally, the management of model "artefacts" (output data, run scripts, configurations, model weights, training protocols, log files) belongs to WP3 and is currently being defined (see draft technical design documents "Artefact infrastructure", "Tracking runs and experiments", and "Metadata and STAC database" in the private repository accessible to project partners only).

The document introduces the guidelines and roadmap for the WeatherGenerator-Atmosphere. Since the team developing the WeatherGenerator-Land is smaller and there are no interdependencies with other partners, simpler procedures have been set in place. These are elaborated in more detail in section 7.

2.2.3 List of design documents produced thus far

The list below contains the current design documents available at the private OpenOffice repository (accessible to project partners only) related to model development. There are also documents on data management, which are not listed here.

- Satellites channels
- Evaluation and diagnostics
- Evaluation tools
- Analysis-ready output data from intermediate representation
- HPC naming conventions
- Structure of git repo coding practices
- Config
- Tracking of training
- Tracking runs and experiments
- Work iterations
- Data readers
- Artifact infrastructure
- Model storage architecture
- Kerchunk and Icechunk-zarr style reference generator and data reader

2.2.4 Deviations and counter measures

No deviations have been encountered.

3 Software repository structure and contribution policy

The WeatherGenerator-Atmosphere code is developed as Open Source code in the central GitHub repository https://GitHub.com/ecmwf/WeatherGenerator. This repository contains the complete software of the core model, including data loaders, network architecture, training procedures, etc. Excluded from this repository are input and output datasets and system-specific run scripts which require adaptation to the individual hardware systems on which the code has been tested. For this, the following "private" repository has been established in Jülich at https://gitlab.jsc.fz-juelich.de/esde/WeatherGenerator-private. All work from other institutes (CSCS, DKRZ, Bologna) happens in public forks on GitHub.

The core development team of the WeatherGenerator has adopted an open contribution policy, the rules of which are described in https://GitHub.com/ecmwf/WeatherGenerator/blob/develop/CONTRIBUTING.md. Since March 2025 several contributors from related projects (e.g., the German research projects RAINA and HClimRep) have joined the developer team and are actively developing code features.

The WeatherGenerator code is structured into the following folders:

- assets: any general material related to the repo, currently only logos
- config: default configuration (yaml file), configurations for data streams and profiling
- integration_tests: test configurations for functional testing on different platforms
- packages: uv subpackages for evaluation code and non-gpu-dependent code shared between evaluation and training/inference.
- scripts: shell scripts to support GitHub workflows
- src/weathergen: the actual WeatherGenerator source code, structured into subfolders datasets, model, train, and utils
- stac: files to build STAC catalogues from WeatherGenerator data
- tests: (rudimentary) unit tests

3.1 Branches

Until the first release of the WeatherGenerator model, the main branch of the repository is `develop`. This branch is always kept up-to-date and contains a model version that is ready to be used, which means that new features are only added to this branch after testing and code review. The development process follows the standard trunk methodology with fast iteration on short-lived branches that are merged within 2-3 weeks. For larger development tasks (for example, adding diffusion models), separate sub-packages have been created in the code base to give relative autonomy.

We use the following patterns for naming branches: `GitHub_username/target_branch/keyword`. For example, `tjhunter/develop/23_linter`. Hence, this branch refers to the pull request associated with Issue #23 (applying the linter rules), implemented by Timothee Hunter.

3.2 Tasks and issues

We use the GitHub tracker, and we use a GitHub project to track work. Locally, other tools (ECMWF JIRA board, Julich JSC Gitlab boards) can be used for site-specific discussions. However, it needs to be ensured that content is transferred to the central GitHub or Wiki once this work is being shared with others.

A set of issue labels has been defined to separate concerns and flag the urgency or status of an issue. The current issue labels are:

- app(lication)
- bug
- datasets
- documentation
- duplicate
- enhancement
- evaluation
- infra
- initiative
- invalid
- model
- quality
- question
- science
- wontfix

Another label is defined as “good first issue” to allow new contributors to identify relatively easy tasks that they can work on to familiarize themselves with the code and software development practices.

4 Coding practices

The core contributors of the WeatherGenerator project come from a diverse background. To facilitate collaboration, a strong set of guidelines has been developed. These guidelines are enforced through automated tool tests.

4.1 Developing joint ways of working

The team has developed joint vocabulary, not only for modelling but also with unexpected issues. For example, we developed a joint naming convention for discussing the various HPCs, realizing that each team had slightly different names (document “HPC naming conventions” in the private repository accessible to project partners only).

We use the `ruff check` formatting tool for the Python language. The set of rules is based on sensible defaults from the industry that have evolved over time. Any failure to adhere to this standard blocks merging of the code.

We use black, as implemented by `ruff format`. It is now the default in the python community and it requires no configuration. This prevents any tedious discussion about various styles.

Issues shall be formulated as small, manageable pieces of work (rule of thumb: max 1 week development time). Once the issue is solved, a test shall be run and a pull request shall be made including a code review to approve of new developments or code changes. If unsure about the content or scope of an issue, authors should open a discussion on Mattermost and they will receive support from the more senior developers.

Compared to industry development teams, the WeatherGenerator team is small. This means that developments often hinge on specific persons rather than sub-teams and this increases the risk of delays due to unavailability of authors or reviewers. Consequently, all team members must learn to make honest assessments about their capabilities and capacities and once committed to a task, dedicate sufficient time to seeing it through within the agreed deadline. It is of utmost importance to communicate timely and to adopt an open error culture that avoids blaming or shaming individuals. Any collaboration issues should be solved

informally rather than setting up complex processes. Where consensus cannot be reached, the Theme coordinator or the project coordinator will be glad to mediate.

4.2 Development of new concepts and larger design changes

As stated above, issues should be specific, small, and manageable. For larger developments and the development of new concepts, contributors are asked to write a design document and announce it in the group for discussion. The discussion will be terminated with a deadline (that can be extended if necessary), and after conclusion is reached, the concept described in the design document is broken into issues that can be tackled individually.

4.3 Dealing with bugs and errors

The current model code has basic sanity tests. This will be expanded once the codebase stabilizes. To maintain the design goal of rapid and uncomplicated uptake by the downstream application developers and others, one or a few standard configurations will be defined and maintained for which end-to-end integration tests with small sample datasets shall be developed, including standard “fast” evaluation routines. However, compared to industrial standards, we have to acknowledge that the WeatherGenerator is a research project, and it is important that sufficient freedom is maintained to explore and iterate fast on ML code and try out new ideas.

4.4 Code reviews

Code reviews are fundamental to strong software foundations. All committed code should be reviewed and approved by another person (ideally from another partner organization).

Changes are done by opening pull requests for review against the develop branch. Direct merging onto the develop branch is disabled. The guidelines below apply to committing against the develop branch. Some developments happen on a fork with a few collaborators; here, the collaborators are free to work as they wish.

The author of a pull request is responsible for getting their work merged. If a reviewer takes longer than expected to review or expresses disagreement with the proposed changes, it is the responsibility of the author of the pull request to proactively resolve this situation. The author can use the dev channel on Mattermost to ask for a review. GitHub offers an option to request a review, but it does not work for people outside the organization (i.e., ECMWF). Asking on Mattermost is therefore strongly preferred.

The reviewer has a responsibility to provide timely, actionable feedback in a courteous and constructive manner. Reviewers (and especially more senior members) should prioritize the review of other people's work over their own work. The Google guidelines for reviewing provide excellent guidance on these aspects.

5 Integration and testing

We generally follow the philosophy of “move fast, correct fast” rather than aiming for perfection in the first attempt. Since we are developing and training a model of unusual complexity, we need to explore quickly the various trade-offs in performance, accuracy and scalability on an end-to-end prototype. This can only be done through quick, measurable experimentation and continuous improvements, including automated workflows that allow for fast evaluation of model results. Such workflows are under development.

6 Release policy

Model releases will be tied to pre-trained model configurations so that immediate uptake of new model versions, for example by the downstream application developers, become possible.

A first prototype release is planned for month 7 of the project (August 2025). This first release version will contain prototype implementations of all relevant model components and workflows. The main objective is to establish a defined starting point for the downstream application development in Theme 3. Note that the scientific quality (i.e., forecast metrics) of this first release will not yet be on par with leading models described in the literature.

Future releases will be planned two months ahead of time. The release target will be defined based on a review of current issues and a discussion among the developers (with user input), which features shall be contained in the release. A release branch will be prepared. This branch will be closed for new developments prior to the release to allow for sufficient testing, bug fixes, and enhanced documentation of the release code. The code release will get a specific tag in the GitHub repository.

7 Deviations for WeatherGenerator-Land

WeatherGenerator-Land is being developed as a research product at the Max Planck Institute for Biogeochemistry. Here, software development guidelines need to strike a good balance between stringency and ease of use. For WeatherGenerator-Land this means in general the guidelines for WeatherGenerator-Atmosphere are adopted, but with a few deviations listed hereafter, which enable quicker development within a much smaller team.

WeatherGenerator-Land adopts a similar repository structure to its atmospheric counterpart, yet excluding some functionality like uv-subpackages, integrations-tests and stac catalogues. For experiment and artifact tracking, WeatherGenerator-Land relies on the open-source `dmlcloud` library which can be found at https://GitHub.com/sehoffmann/dmlcloud . A CI/CD workflow is implemented via GitHub Actions. WeatherGenerator-Land also adopts private repositories for system-specific run scripts and configurations.

WeatherGenerator-Land will also make use of GitHub Issues and Pull Requests for managing code development, but the concrete guidelines are understood less strictly: 1. branch naming and issue categories are more free, 2. code reviews are understood as good practice, but it is acceptable to merge smaller changes, if this enables quicker development, 3. design documents are not obligatory for new features, but encouraged and new features are supposed to be discussed during regular developer meetings.

The release policy of WeatherGenerator-Land is different from WeatherGenerator-Atmosphere and will follow closely the milestones and deliverables in the grant agreement.

8 Conclusion

In this document we have summarized the software development procedures and practices and described the tools that have been set up to structure and guide the WeatherGenerator development. As described in the individual sections above, some aspects of the “model development infrastructure” are still under development, but by and large, the team has established well-working practices which balance the needs for code stability, reproducibility, portability, and efficiency that are required by the downstream application developers and

other users with the flexibility that is needed to quickly test new scientific ideas or explore new technical concepts.

The rules and guidelines described here have been adopted by all contributing partners and have proven quite efficient and successful. It is noteworthy, however, that the development of a scientific model like the WeatherGenerator not only requires good tools, but also a highly motivated team of skilled software engineers and scientists. Through the organisation of regular (virtual) meetings and a series of hackathons, we managed to form a strong team and therefore prospects are good that the ambitious goals of the core model development can be reached.

Document History

Version	Author(s)	Date	Changes
0.1	Martin Schultz, Tim Hunter, Christian Lessig	16/06/2025	Initial version
0.2	Contributions by Florentine Weber, Savvas Melidonis, Ankit Patnala and Michael Langguth	25/06/2025	Added comments and details
0.3	Martin Schultz	02/07/2025	Consolidated version, resolving most comments and adding information on general collaboration practices and issue labels
0.4	Michael Langguth, Christian Lessig, Vitus Benson, Sebastian Hoffmann	09/07/2025	Final review by co-authors, missing information added
0.9	Martin Schultz	09/07/2025	Merged all changes and submitted for internal review
1.0	Martin Schultz	21/07/2025	Finalized document after internal review

Internal Review History

Internal Reviewers	Date	Comments
Julian Kuehnert	15/07/2025	Consistency and editorial content
Marieke Wesselkamp	18/07/2025	Consistency of naming and formatting