SoftwareX 34 (2026) 102630

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

GEOMODELATOR-GUI: A web-based graphical user interface for 3D geological modelling

Thomas Kempka [a,b,*], Benjamin Nakaten [a], Elena Chabab [a]

[a] GFZ Helmholtz Centre for Geosciences, Telegrafenberg, 14473 Potsdam, Germany
[b] University of Potsdam, Institute of Geosciences, Karl-Liebknecht-Str. 24-25, 14476 Potsdam, Germany

ARTICLE INFO

Keywords:
Geosciences
Geological modelling
Structural data
Web application
Graphical user interface
Python

ABSTRACT

Geological modelling workflows often rely on command-line tools requiring programming expertise, limiting accessibility for domain scientists. This paper introduces GEOMODELATOR-GUI, a novel web-based graphical user interface for the open-source 3D geological modelling software GEOMODELATOR. The GUI abstracts complex configuration and execution processes into an intuitive, step-wise wizard interface, enabling geoscientists to generate structural geological models without coding, and thereby turning available field data into practical insights in a straightforward process. Built using Streamlit for the client and Flask for the server, the application implements a session-based architecture where model configurations are transmitted via JSON. The core innovation lies in its hybrid local-storage approach: large geological structural datasets and model outputs (ParaView VTK and VTS, numpy npz) are managed via a shared directory to circumvent network transfer limitations, while configuration logic remains server-mediated. The system performs model pre-configuration including grid alignment and data transformation, partitioning using cubic interpolation and priority-based sequential assignment of fault/strata relationships, and post-processing by inverse transformation and output generation. Validation using synthetic and real-world datasets demonstrates successful generation of complex faulted and stratified geological models. GEOMODELATOR-GUI significantly lowers the barrier to entry for geological modelling, enhances reproducibility through session management, and provides interactive 3D visualisation via PyVista. The fully open-source Python implementation of the GUI supports efficient 3D model development for numerical simulations within short time frames.

Metadata

Nr	Code metadata description	Metadata
C1	Current code version	v1.0
C2	Permanent link to code/repository used for this code version	https://github.com/bnakaten/geomodelatorgui
C3	Permanent link to reproducible capsule	https://doi.org/10.5880/GFZ.AJPH.2025.002
C4	Legal code license	GNU General Public License, Version 3, 29 June 2007
C5	Code versioning system used	git
C6	Software code languages, tools and services used	Python, Streamlit, Flask, PyVista, stpyvista, NumPy, SciPy, gdal, geos
C7	Compilation requirements, operating environments and dependencies	Python 3.9+, see environment.yml for complete dependency list
C8	If available, link to developer documentation/manual	https://git.gfz.de/bnakaten/geomodelator/-/wikis/home
C9	Support email for questions	benjamin.nakaten@gfz.de

1. Motivation and significance

Geological modelling represents a fundamental component in resource exploration, subsurface utilisation assessment and engineering projects. Despite the availability of numerous 3D geological modelling tools, many require acquiring commercial closed-source software or significant programming expertise when implemented as command-line interfaces, which hinders accessibility and creates substantial barriers for scientists who lack coding proficiency. GEOMODELATOR, an open-source Python library for generating structural geological models from fault and stratigraphic data, originally operated exclusively via command-line interface, necessitating manual adaptation of Python scripts for each model configuration. This workflow proved error-prone, lacked interactivity, and hindered broader adoption by users with

* Corresponding author at: GFZ Helmholtz Centre for Geosciences, Germany.
E-mail address: thomas.kempka@gfz.de (T. Kempka).

https://doi.org/10.1016/j.softx.2026.102630
Received 24 November 2025; Received in revised form 16 March 2026; Accepted 26 March 2026
Available online 31 March 2026

domain expertise but limited programming skills.

The development of GEOMODELATOR-GUI addresses these critical limitations by providing an intuitive web-based interface that encapsulates GEOMODELATOR's core functionality within a guided, wizard-driven environment. Unlike existing solutions, GEOMODELATOR-GUI uniquely combines accessibility through form-based configuration with the ability to handle large and complex geological datasets through its hybrid local-storage architecture. This approach significantly lowers the technical barrier to entry while maintaining the computational rigour required for professional geological modelling. Hereby, users with limited-memory workstation systems, e.g. laptops, can benefit from GEOMODELATOR-GUI's hybrid architecture, allowing to transfer memory-intensive tasks to computing systems offering the required memory capacity.

In the revised setting, users interact with GEOMODELATOR-GUI through a web browser where they progressively define model parameters including grid dimensions, structural geological data, and configuration priorities. The software then processes these inputs to generate comprehensive 3D geological models with partitioned fault and stratigraphic relationships. Related work in geological modelling includes different open-access and commercial tools, which offer varying degrees of user-friendliness but often require extensive programming skills. GEOMODELATOR-GUI distinguishes itself by providing a completely open-source solution with an intuitive interface that requires no coding, making sophisticated geological modelling accessible to a wider scientific community for use in a variety of applications.

The original GEOMODELATOR was developed as a stand-alone, locally usable command-line application where configuration occurred through adapting Python code according to model requirements, supplemented by geological structural strata and fault data files as 3D coordinate ASCII files (CSV, data structure example: https://github.com/bnakaten/geomodelatorgui/blob/master/static/demo/layer-1.vtk) [1,2]. These input data can be generated and pre-processed in different available open-source tools, i.e. FreeCAD [3]. Execution of GEOMODELATOR produced final model grids (ParaView VTS files, data structure example: https://github.com/bnakaten/geomodelatorgui/blob/master/static/demo/model.vts), partition data (Python Numpy [4] npz data), and structural strata and fault surfaces (ParaView [5] VTK files, data structure example: https://github.com/bnakaten/geomodelatorgui/blob/master/static/demo/fracture1.vtk). This command-line GEOMODELATOR application code has been restructured and extended into a stand-alone, class-based Python library "gml.py" that contains the definition of the central "Model" class. Through this class, the model, geological structural strata and fault data, can be successively read in or imported as a complete JSON string [6] Additionally, a masking file containing a list of coordinates can be used to selectively ignore specific areas of the model during processing, e.g. for user-defined model boundaries [7].

A brief overview of the GEOMODELATOR-GUI key features compared to those of the currently most applicable open-source geological modelling software packages GemPy [8] and LoopStructural [9] is provided in Table 1. While the existing software packages currently offer more refined modelling methods, GEOMODELATOR-GUI benefits from its workflow accessibility for non-programmers and ability to outsource computationally heavy tasks to dedicated servers with the GUI running on a low-memory workstation or laptop.

2. Software description

GEOMODELATOR-GUI implements a client-server architecture comprising two primary components: (1) a Streamlit-based [10] client for user interaction and visualisation, (2) a Flask application managing model configuration and execution, using a shared local storage directory (e.g. realised via nfs, samba, sshfs or WebDAV) for handling large geological datasets (Fig. 1). This hybrid architecture addresses a fundamental challenge in web-based geological modelling, namely the impracticality of transferring multi-gigabyte 3D structural datasets over HTTP by means of a shared file system while retaining the flexibility of server-side processing, as documented by [2].

The server is implemented using Flask [11], a lightweight WSGI web application framework, and manages two primary communication routes. The POST route (/configure) receives model configuration via a JSON string, validates it for syntactic correctness, and stores it for the

Table 1
Overview of GEOMODELATOR-GUI features in comparison with the open-source geological modelling frameworks GemPy [8] and LoopStructural [9].

	GEOMODELATOR-GUI	LoopStructural	GemPy
GUI	**Yes** (Web-based interactive interface including a modelling wizard)	**Limited** (Python API with built-in visualisation tools like LavaVu and matplotlib)	**Limited** (Python API; VTK-based interactive 3D viewer available via command, not a dedicated UI)
Modelling methods	Implicit modelling (lattice/grid based). Uses cubic interpolation on coordinate planes for surfaces and sequential partitioning for fault/strata relationships based on user-defined priorities.	Implicit modelling. Offers a variety of interpolation methods including discrete approaches (piecewise linear on tetrahedral mesh, finite-difference on Cartesian grid) and data-supported approaches (Radial Basis Function, Dual Co-Kriging).	Implicit modelling. Based on the potential-field method using universal Co-Kriging (dual Co-Kriging interpolation).
Required programming skills	**None to low**. Designed for geoscientists with limited programming skills; uses a point-and-click web wizard for configuration and operation.	**Moderate to high**. Requires knowledge of Python to use the API as a library for model definition and scripting workflows.	**Moderate to high**. Requires knowledge of Python to use the API; designed as a library for coupling with Python-based scientific workflows.
Client-server based architecture	**Yes**. Implements a client-server architecture using Flask (server) and Streamlit (client) with a hybrid local-storage approach for large datasets.	**No**. Implements a local, session-based architecture within a Python environment (e.g., Jupyter notebooks).	**No**. Implements a local architecture using Python libraries (Theano, NumPy, etc.). Theano allows client-side (CPU/GPU) optimised execution.
Data import and export	Accepts geological structural data and masking files (ASCII CSV, GIS-compatible shapefiles). Exports model grids (ParaView VTS), partition data (NumPy npz), and structural surfaces (ParaView VTK).	Data is generally managed via Pandas DataFrames. User can load custom test/reference datasets. Capable of exporting triangulated surfaces (VTK, OBJ) and model evaluation results.	Uses Pandas DataFrames for input data management (interface points and orientations). Supports exporting to common formats (e.g., VTK/Paraview) and visualisation tools.
Key benefits	Offers the **most accessible workflow for non-programmers** due to its client-server architecture and intuitive web interface, specifically bridging the gap between domain experts and complex modelling without requiring coding. Enables outsourcing computationally heavy tasks to dedicated servers, while the GUI runs on low-spec hardware.	Provides the **most versatile modelling framework** with multiple interpolation algorithms (including discrete and data-supported) and time-aware geological concepts, making it highly adaptable for research and complex structural configurations.	Provides the **strongest foundation for uncertainty quantification and probabilistic modelling** (via PyMC) and is optimised for performance using automatic differentiation (Theano), making it ideal for advanced machine learning and Bayesian inference workflows.

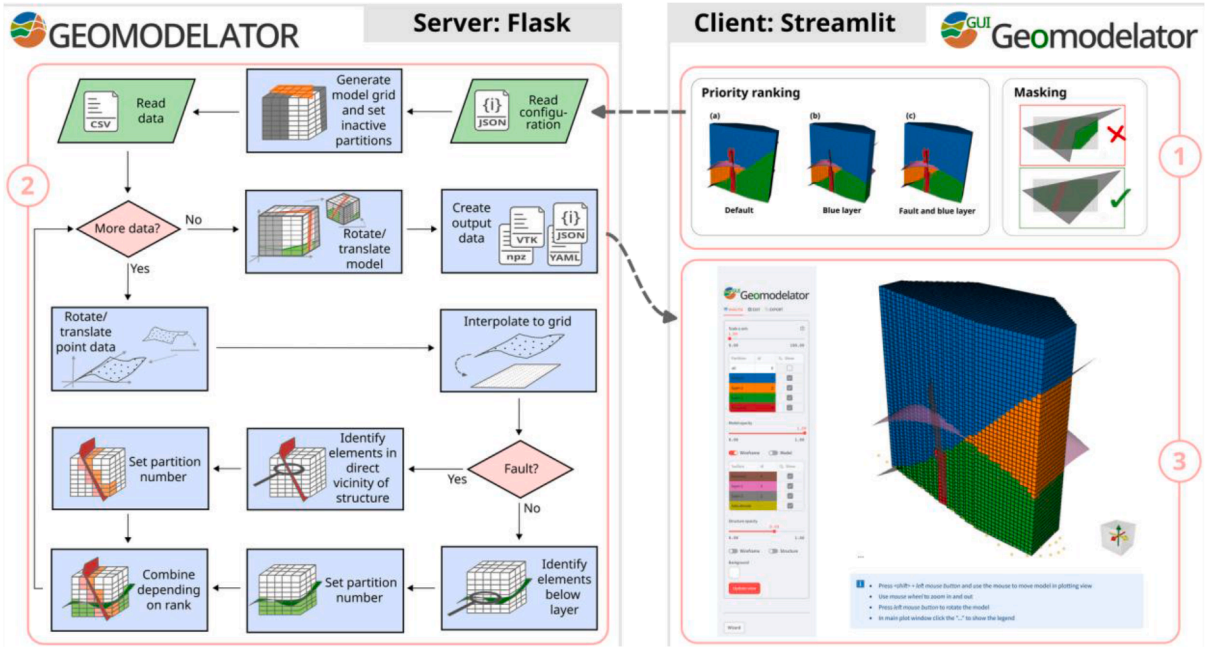


Fig. 1. Hybrid architecture of GEOMODELATOR-GUI based on (1) Streamlit client and (2) Flask server interaction. The model processing workflow dedicated to the Flask server component (2) is visualised in the GEOMODELATOR-GUI (3), which allows for interactive model pre- and post-processing.

current individual request session. Consequently, any user-interaction is updated ad-hoc in the JSON configuration file (example data structure given in: https://github.com/bnakaten/geomodelatorgui/blob/master/static/demo/configuration.json). The GET route (/generate) subsequently initiates the model generation process, executing the complete workflow of model configuration and generation using the session data. After model generation at the file level, these files are transmitted to the client. In the current implementation, files are accessed through a "local" directory to which both server and client maintain access. During client upload and server model generation, this local directory is also accessed for storing the temporary and generated files. The client also retrieves files from this directory for visualisation. While the current version does not support genuine server-client operation over a network, this capability could be implemented with minimal effort. Due to the large file sizes associated with geological structural data and its visualisation, this local directory approach is currently preferred.

The modelling process follows a structured pipeline beginning with pre-configuration, where a rectangular cell-based grid is generated from user-defined corner points, model dimensions, and discretisation parameters. This grid is then translated to the origin and aligned with coordinate planes by translation and rotation, with the translation vector and rotation angle stored for later use. Then, a cell-centred grid is created from this model. The geological structural data, including strata and fault surfaces, are read and transformed using the same translation vector and rotation angle. The standard first-come-first-served structure data ranking is overridden by a user-specified ranking provided in the web-based GUI to account for geological principles. For geological faults, the standard fault width is overwritten by a user-specified value. If provided by the user, a masking file is read and transformed considering the translation vector and rotation angle, and then applied to the model to mark cells as active or inactive in numerical simulations.

The partitioning phase represents the computational core of GEOMODELATOR-GUI. For each coordinate plane (XY, XZ, YZ), the complementary coordinate (Z, Y, X) of the structure datasets is interpolated using scipy.interpolate.griddata [12] with the cubic method and stored in a 2D matrix. This 2D matrix contains both interpolated values and NaN values, because not every structural dataset may occupy the complete dimensions of the model coordinate planes. From this, a ratio between interpolated values and the number of cell centres of the model per coordinate plane is determined. Based on this ratio, the preferred interpolation plane (XY, YZ, or XZ) for partitioning is selected. The preferred interpolation plane provides the data for the visualisation file of the structure data (VTK).

Subsequently, sequential partitioning occurs. For fault structures, the cell-centre model is traversed cell by cell, and the cell-centre coordinates are compared with the preferred interpolated coordinate values. Smaller values lie on one side of the structure and are marked accordingly with a specific partition ID. Larger values lie on the other side and are marked with a different ID. This allows cells positioned to the left and right of the specific structure to be identified accurately during the later processing. Using the user-provided structure's priority ranking, it is checked whether the partition data should be recorded in the result matrix. If so, the fault width is checked, and the marking is only adopted if the absolute distance between the model cell centre and the fault interpolation value is less than the fault width.

For geological strata, the cell-centre model is traversed, and the cell-centre coordinates are compared with the preferred interpolated coordinate values. Smaller values lie on one side of the structure and are marked accordingly with a distinct partition ID. This allows cells positioned above and below the structure to be identified in the same way as for the geologic fault structures. Using the structure's priority ranking, it is checked whether the partition data should be recorded in the result matrix.

The post-configuration phase transforms the partitioned model back to its original position by inverse rotation and translation. From the partitioned model, the 3D partition matrix is written as a multi-dimensional numpy-array to an npz file. On the basis of the model grid and the 3D partition matrix, the output file (ParaView VTS) is created.

The client-side GUI is implemented in Streamlit and provides direct access to the two essential functions: (1) model representation and (2) model configuration. During model configuration, model data is entered step by step in a sequential wizard format. Alternatively, an example model configuration can be used, modified, and subsequently generated into a model. The model representation is interactive through stpyvista and positioned centrally in the main window. A navigation and

configuration sidebar offers ANALYSIS, EDIT, and EXPORT options. Using the ANALYSIS tools, the iterative representation of the model can be exaggerated vertically, and layers/partitions and surfaces can be toggled on and off. The EDIT tool allows all configuration data to be reviewed and manipulated. For editing, the wizard is called with the corresponding configuration step. The EXPORT tool provides the configuration file as well as all model data in a compressed ZIP file for download.

For communication with the server, a request session is used that both sends the model configuration data via HTTP POST to the server and requests model generation (partitioning) via HTTP GET.

3. Illustrative examples

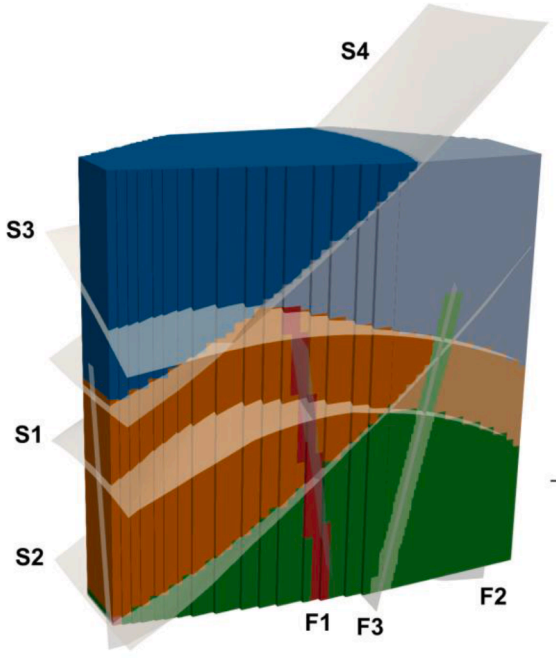
The first application case is introduced to demonstrate how GEOMODELATOR-GUI can be applied to represent structural geological features with different priorities in a simplified geological model, while the second one introduces a complex geological model derived from real-world geological data from the Northeast German Basin region.

Fig. 2 illustrates GEOMODELATOR-GUI's stratigraphic and fault surface ranking methodology, and provides a representative visual example of the high accuracy achievable by the applied cubic interpolation method for a model with four stratigraphic layer and three fault surfaces. The stratigraphic layer surface S4 is assigned with the highest rank, so that it divides the model into the dark blue partition and those below. Grid elements situated between S4 and S3 (rank 4) are assigned to the light-blue partition. The top of the orange partition is defined by S4 and S3, while that of the light-orange partition is set by S3 and S2 (rank 3), while S2 and S1 (rank 1) determine that of the green partition. Fault surfaces F3 (rank 7) and F1 (rank 6) define the light green and red partitions, respectively, and extend without limitation through the partitions determined by S2, S1 and S3 due to their higher ranks. F2 is concealed within the green partition due to its low rank (2), visible only as a single pink cell in the lower left corner of the model. At its intersections with F1 and F3, the partition defined by F2 is interrupted, as the constituent grid elements already belong to partitions defined by higher-ranked stratigraphic and fault surfaces.

The modelling of different geological scenarios is presented in Fig. 3 by means of simplified geological models. An erosional scenario is illustrated in Fig. 3(a) with the stratigraphic layer surface S1 - possessing the highest rank - truncating the partitions defined by S2 and S3. A channel erosion scenario is shown in Fig. 3(b) with S5 representing the base of the erosional channel, cutting through all other partitions from above due to its highest rank. Fig. 3(c) focuses on the implementation of fault surfaces, where all faults possess higher ranks than the stratigraphic surfaces. As the fault surfaces do not intersect each other, their ranks are not of further relevance here. If the faults would have lower ranks than the stratigraphic surfaces, they would not define separate partitions and would not be visible as dedicated grid elements in the model. The light-green fault partition in Fig. 3(d) described by F1 holds the highest rank, and is accordingly not bounded at the model top and bottom. In contrast, F2 has a lower rank than S1 and S2, and consequently the partition assignment ceases at the top partition bounded by S2. F2 and F3 have lower priorities relative to F1 due to their lower ranks, and hence end as they intersect F1.

The geological model presented in Fig. 4 was developed to address hydrogeological research questions within a sub-area of the Lower Spree catchment area in the Federal State of Brandenburg in Germany. The model incorporates 16 stratigraphic layers and multiple geological fault structures, representing a complex geological setting typical of rift systems. Users begin by launching the application through standard commands for the Streamlit client and Flask server. Within the browser interface, the configuration wizard guides the user through defining the model grid dimensions (190 × 115 × 290 cells covering a 58 km x 35 km x 870 m volume), uploading geological structural data files based on spatial point cloud datasets containing XYZ coordinates in ASCII CSV format, and specifying structure priorities. These data is used to interpolate the stratigraphic surface geometries and delineate the boundaries of individual geological layers as well as fault zones within the model, whereby for each fault zone an individual thickness parameter can be assigned to accurately reflect local geological variability.

A masking file can be imported via external data sources such as ASCII CSV files containing XY coordinates or GIS-compatible shapefiles to define user-specified irregular model boundaries in the modelling



Structure	Rank
F3	7
F1	6
S4	5
S3	4
S2	3
F2	2
S1	1

Fig. 2. Application example visually demonstrating the good accuracy of the chosen cubic interpolation method used in GEOMODELATOR-GUI, applied to a simplified geological model with four stratigraphic layer and three fault surfaces. The rank of the stratigraphic layer and fault surfaces defines the final model partitioning and fault representation in the model, and is provided in tabular form inline.

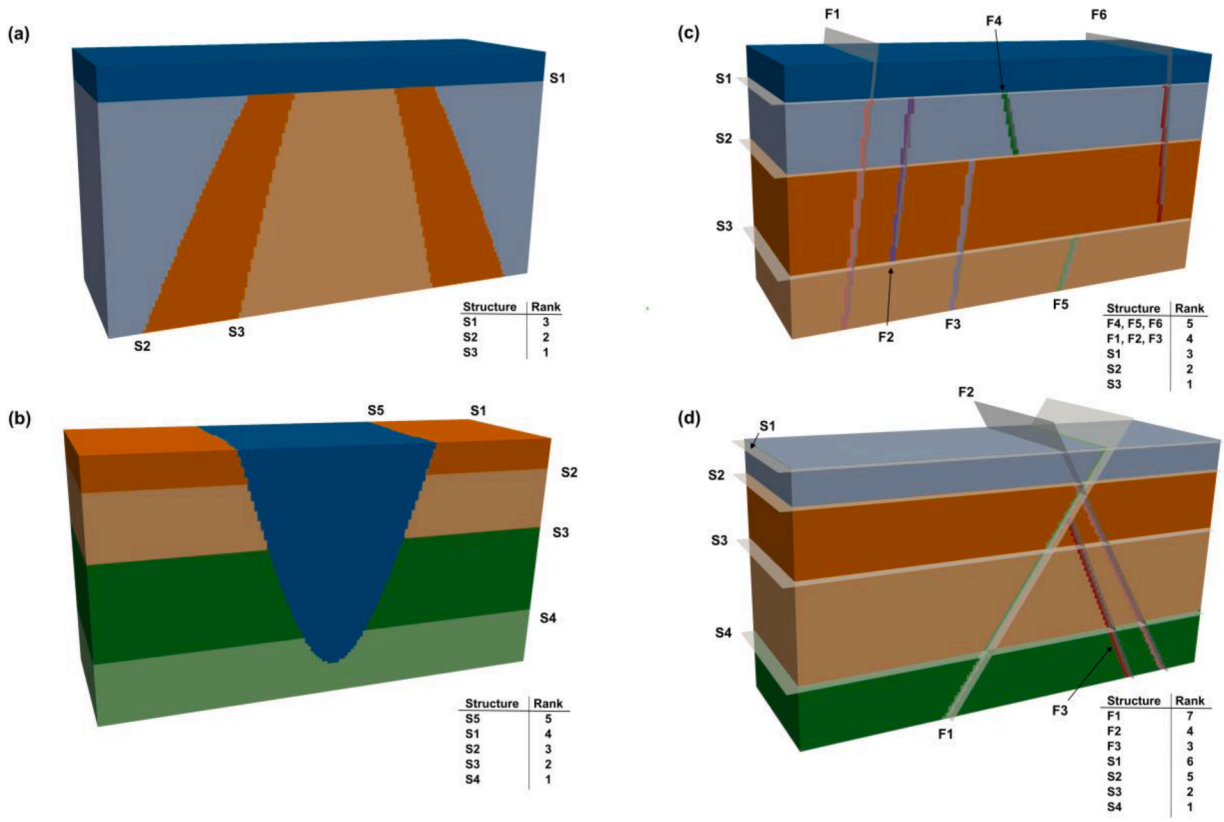


Fig. 3. Application examples detailing the role of stratigraphic and fault surface ranking in geological settings including (a) an erosional stratigraphic layer, and (b) an erosional channel as well as the methodology of ranking fault surfaces to assign geological faults to specific partitions (c), and orchestrate their behaviour at fault surface intersections (d). The ranking of the specific stratigraphic and fault surfaces for each example is provided in tabular form inline.

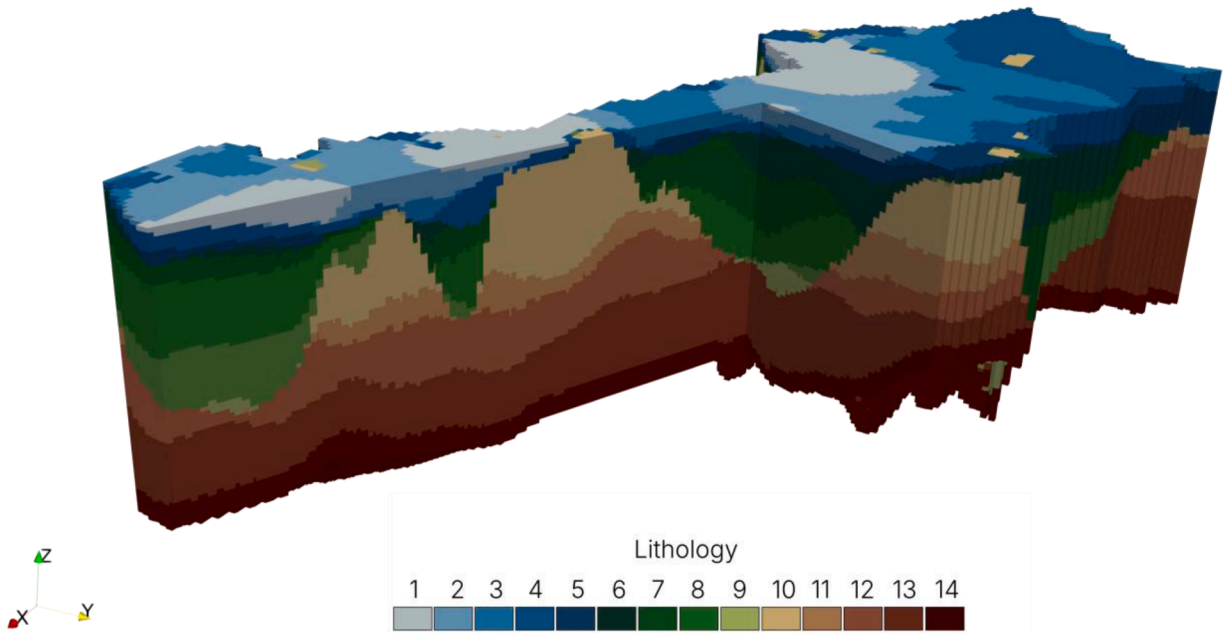


Fig. 4. GEOMODELATOR-GUI generates outputs in VTS and VTK file formats that enable comprehensive visualisation, interpretation, and analysis of the structural geological data, facilitating further understanding of spatial relationships and model characteristics.

process, ensuring that the relevant regions can be properly addressed in later numerical simulations. After completing the configuration steps, the user initiates model generation through the interface. The system processes the configuration in <2 min on standard laptop hardware, i.e. with Intel Core™ i7–1355 U CPU and 32 GB LPDDR5 SDRAM, performing the complete workflow from grid alignment through partitioning to output generation.

Upon completion, the interactive 3D visualisation displays the

resulting model with distinct colours representing different stratigraphic units. Using the ANALYSIS tools, the user can apply a vertical exaggeration to visualise subtle structural features and selectively hide individual layers or fault structures to examine the underlying stratigraphy. The EDIT functionality allows inspection and modification of any configuration parameter, with the wizard interface automatically restoring the relevant configuration step. Finally, the user exports the complete model package, which includes all input parameters, geological data, and generated outputs in standard formats (ParaView VTS and VTK, numpy-npz), ensuring full reproducibility of the modelling workflow.

The finalised hydrogeological model exemplarily shown in Fig. 4 represents a complex 3D system that encompasses the geological features and bedding characteristics, such as glacial erosion channels, stratigraphic discontinuities, and multiple distinct fault zones. Thereby, it is demonstrated how GEOMODELATOR-GUI enables geoscientists to generate and analyse sophisticated geological models without programming expertise. This transforms a process that previously required hours of programming and debugging into a streamlined workflow completed in minutes. The exported model package can be directly imported into standard visualisation tools like ParaView for further analysis or shared with collaborators to ensure methodological transparency. The masking functionality proves particularly valuable in this case, as it allows consideration of hydrogeological boundary conditions in the modelling domain, while maintaining grid continuity in surrounding regions. Further application examples are presented by [13–20] and [21].

4. Impact

GEOMODELATOR-GUI has significantly impacted geological modelling practices by its open-source access to sophisticated 3D modelling capabilities. Prior to its development, generating structural geological models required either commercial software licenses or programming expertise to adapt command-line tools like the original GEOMODELATOR implementation. This created a substantial barrier for geoscientists whose primary expertise mainly lies in geological interpretation rather than software development. By replacing code editing with an intuitive configuration wizard, GEOMODELATOR-GUI has reduced model setup time from hours to minutes, allowing researchers to focus on geological interpretation as a preparatory step to numerical simulations rather than implementation details.

The software has enabled new research questions regarding rapid scenario testing in geological modelling, as researchers can now efficiently generate multiple model variants with different structural assumptions. Within the GFZ Potsdam, the software has become integral to projects using coupled numerical simulation to investigate subsurface utilisation in view of its efficiency and environmental impacts, where rapid model generation supports decision-making. The session-based architecture and export functionality have enhanced research reproducibility, with complete modelling workflows now easily shared among collaborators and documented in publications.

GEOMODELATOR-GUI has changed daily practice for geoscientists by providing immediate visual feedback during model configuration, allowing for iterative refinement based on emerging geological insights. This interactive approach contrasts sharply with traditional command-line workflows where visualisation occurred only after lengthy processing procedures had been completed. The masking functionality has proven particularly valuable for geoscientific projects, where maintenance of relevant model boundaries improves computational efficiency and geological relevance.

The hybrid storage architecture represents a novel approach to handling large and complex geological datasets in web applications, potentially influencing future development of scientific web tools facing similar data size challenges. By demonstrating that complex geological modelling can be made accessible through thoughtful and targeted interface design, GEOMODELATOR-GUI contributes to the broader movement toward providing scientific computing tools across disciplines. The software has already been integrated into teaching activities at the University of Potsdam, where it serves as an educational tool for introducing students with limited programming prerequisites to geological modelling and numerical simulation concepts.

5. Conclusions

GEOMODELATOR-GUI successfully transforms GEOMODELATOR from a command-line tool requiring programming expertise into an accessible platform for geoscientists. The web-based interface significantly lowers technical barriers to 3D geological modelling while maintaining computational rigour through its partitioning algorithms and structured workflow. The hybrid local-storage architecture effectively addresses challenges associated with large geological datasets, representing an effective solution for scientific web applications.

The software's impact is evident in reduced model setup time, enhanced reproducibility through session management, and increased accessibility for domain scientists. By focusing on user experience without compromising scientific validity, GEOMODELATOR-GUI exemplifies how thoughtful software design can bridge the gap between sophisticated algorithms and end-user needs.

Future development will focus on cloud storage integration to enable true client-server operation over networks, advanced uncertainty quantification capabilities, and expanded integration with other geological modelling frameworks. The open-source nature of GEOMODELATOR-GUI ensures ongoing community contributions and adaptation to emerging research needs, promising continued evolution as a valuable tool for the geoscientific community. The software represents a significant step towards making 3D geological modelling accessible to all geoscientists, regardless of their programming background, thereby accelerating geological model development and interpretation.

CRediT authorship contribution statement

Thomas Kempka: Writing – original draft, Supervision, Software, Project administration, Methodology, Investigation, Conceptualization. **Benjamin Nakaten:** Writing – review & editing, Visualization, Software, Methodology, Investigation, Conceptualization. **Elena Chabab:** Writing – review & editing, Visualization, Validation, Methodology, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the GEOMODELATOR user community for valuable feedback during development and the GFZ Computing Centre for infrastructure support. Further, the authors would like to express their gratitude to the involved anonymous reviewers for their efforts and constructive comments.

References

[1] Nakaten, B., 2025a. GeomodelatorGUI - web-based graphical user interface for Geomodelator, developed using Streamlit and Flask. https://doi.org/10.5880/GFZ.AJPH.2025.002.

[2] Nakaten, B., 2025b GEOMODELATOR - A python library to generate simple structured 2.5D and 3D cell-based VTK models/files with layer, fault and seams as corresponding element/zone groups for further processing or integrating into a simulator framework (TRANSPORTSE). https://doi.org/10.5880/GFZ.AJPH.2025.001.

[3] Juergen, R., Mayer, W., van Havre, Y., 2001. FreeCAD. FreeCAD.

[4] Harris CR, Millman KJ, Walt van der SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, Kerkwijk MHV, Brett M, Haldane A, Río JFD, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE. Array programming with NumPy. Nature 2020;585:357–62. https://doi.org/10.1038/s41586-020-2649-2.

[5] Ayachit U. The paraview guide: updated for paraview version 4.3, full color version. Clifton Park, NY: Kitware Inc; 2015. editor.

[6] Pezoa F, Reutter JL, Suarez F, Ugarte M, Vrgoč D. Foundations of JSON schema. In: Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee; 2016. p. 263–73.

[7] Nakaten, B., Chabab, E., Kempka, T., 2025. GEOMODELATOR reloaded – now with GUI and client-server architecture. https://doi.org/10.5194/egusphere-egu25-10442.

[8] De La Varga M, Schaaf A, Wellmann F. GemPy 1.0: open-source stochastic geological modeling and inversion. Geosci. Model Dev 2019;12:1–32. https://doi.org/10.5194/gmd-12-1-2019.

[9] Grose L, Ailleres L, Laurent G, Jessell M. LoopStructural 1.0: time-aware geological modelling. Geosci. Model Dev 2021;14:3915–37. https://doi.org/10.5194/gmd-14-3915-2021.

[10] Sullivan C, Kaszynski A. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). JOSS 2019;4:1450. https://doi.org/10.21105/joss.01450.

[11] Grinberg M. Flask web development. 2nd ed. O'Reilly Media, Inc; 2024.

[12] Virtanen P, Gommers R, Oliphant TE, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods 2020;17:261–72. https://doi.org/10.1038/s41592-019-0686-2.

[13] Chabab E, Kühn M, Kempka T. Upwelling mechanisms of deep saline waters via quaternary erosion windows considering varying hydrogeological boundary conditions. Adv. Geosci 2022;58:47–54. https://doi.org/10.5194/adgeo-58-47-2022.

[14] Kempka T, Kühn M. Numerical simulation of spatial temperature and salinity distribution in the Waiwera geothermal reservoir, New Zealand. Grund. - Z. Fachsekt. Hydrogeol. 2023;28:243–54. https://doi.org/10.1007/s00767-023-00551-8.

[15] Li Z, Chabab E, Spangenberg E, Schicks JM, Kempka T. Geologic controls on the genesis of the Arctic permafrost and sub-permafrost methane hydrate-bearing system in the Beaufort–Mackenzie Delta. Front. Earth Sci. 2023;11.

[16] Li Z, Spangenberg E, Schicks JM, Kempka T. Numerical simulation of hydrate formation in the LArge-scale reservoir simulator (LARS). Energies 2022;15:1974. https://doi.org/10.3390/en15061974.

[17] Li Z, Spangenberg E, Schicks JM, Kempka T. Numerical simulation of coastal sub-permafrost gas hydrate formation in the Mackenzie Delta. Can. Arct. Energ. 2022;15:4986. https://doi.org/10.3390/en15144986.

[18] Otto C, Steding S, Tranter M, Gorka T, Hámor-Vidó M, Basa W, Kapusta K, Kalmár I, Kempka T. Numerical analysis of potential contaminant migration from abandoned In situ coal conversion reactors. Adv. Geosci 2022;58:55–66. https://doi.org/10.5194/adgeo-58-55-2022.

[19] Schnepper T, Kühn M, Kempka T. Effects of permeability and pyrite distribution heterogeneity on pyrite oxidation in flooded lignite mine dumps. Water (Basel) 2025;17:3157. https://doi.org/10.3390/w17213157.

[20] Steding S, Kempka T, Kühn M. How insoluble inclusions and intersecting layers affect the leaching process within potash seams. Appl. Sci. 2021;11:9314. https://doi.org/10.3390/app11199314.

[21] Steding S, Kempka T, Zirkler A, Kühn M. Spatial and temporal evolution of leaching zones within potash seams reproduced by reactive transport simulations. Water (Basel) 2021;13:168. https://doi.org/10.3390/w13020168.