SeisAgentBench: A Failure-Aware Core Benchmark for Multi-Agent Earthquake Rupture Inversion and Forward Validation

Takayuki Shinohara [1]

Abstract

Earthquake rupture studies require coordinating preprocessing, source exploration, finite-fault inversion, and physics-based forward validation across heterogeneous tools. We present **SeisAgentBench**, a failure-aware core benchmark for evaluating multi-agent orchestration over **Grond**, **WISP**, **OpenSWPC**, **SeisSol**, **SPECFEM**, **SW4**, **gmprocess**, **ObsPy**, and **Pyrocko**. Standardized interaction episodes record scientific intent, preprocessing state, candidate models, validation outcomes, failure categories, and provenance for reproducible comparison. **GeoGPT** is included only as an optional supplemental evaluator artifact rather than as ground truth or a replacement for deterministic solvers. By fixing the initial release to rupture inversion plus forward validation, the benchmark provides an auditable testbed for disagreement handling, reranking, abstention, and reproducible execution in AI-assisted seismology.

1. Introduction and Scientific Bottleneck

Earthquake rupture-process inversion remains difficult because fault-slip histories must be inferred under strong uncertainty in structure, Green's functions, regularization, and data selection (Minson et al., 2013). The community has long recognized non-uniqueness, which is why initiatives such as Source Inversion Validation and related standardization efforts became important for comparability and reproducibility (Mai et al., 2016b). In short, progress depends not only on stronger inversion algorithms, but also on interoperable workflows that preserve assumptions and context.

In practice, rupture-analysis studies already depend on multiple open tools spanning preprocessing, source exploration, inversion, and forward simulation (Krischer et al., 2015; Heimann et al., 2017; 2018; Goldberg et al., 2026; Maeda et al., 2017). ObsPy, Pyrocko, and gmprocess support waveform conditioning and event analysis; Grond and WISP support source exploration and finite-fault inversion; and OpenSWPC, SeisSol, SPECFEM, and SW4 support high-fidelity forward validation. However, these systems differ in formats, preprocessing conventions, and execution assumptions. Researchers therefore spend considerable effort translating data and checking whether candidate models remain comparable across tool boundaries.

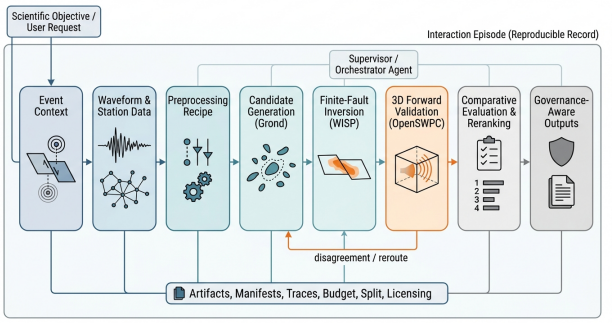

Figure 1. Conceptual overview of SeisAgentBench. Episodes connect preprocessing, inversion, forward validation, failure labeling, and governance-aware outputs for reproducible AI-for-Science evaluation.

This fragmentation is a direct bottleneck for AI-assisted seismology. Tool-using agents can plan and invoke software steps, but there is still no widely adopted resource for training and evaluating agents on realistic rupture-analysis orchestration (Liu et al., 2024; Chen et al., 2024; Starace et al., 2025). Without a shared benchmark, observed gains are difficult to interpret because improvements may come from hidden preprocessing or backend defaults rather than better scientific reasoning. We target this gap with **SeisAgentBench**, a failure-aware core benchmark centered on standardized interaction episodes that connect hypothesis generation, inversion, forward validation, and explicit handling of disagreement and operational failures (Mai et al., 2016a).

Submitted to the AI for Science workshop (ICML 2026).
[1]National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan. Correspondence to: Takayuki Shinohara <shinohara.takayuki@aist.go.jp>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

2. Core Benchmark Roadmap

As illustrated in Fig. 1, each benchmark episode links data preparation, inversion, validation, and governance-aware outputs in a reproducible record.

We define the benchmark unit as an *interaction episode*, not a single input-output pair. Each episode captures event definition, objective, waveform/station context, preprocessing recipe, backend actions, candidate outputs, validation results, and execution traces. It functions as a case file recording what was asked, what evidence was used, what computations were run, and how conclusions were reached. This framing matches real rupture analysis, which is staged and iterative rather than one-shot prediction, and preserves intermediate decisions for auditability.

The acquisition plan is phased. **Phase 1** releases synthetic and semi-synthetic inversion/validation episodes with controlled source geometry and structure for high-quality early benchmarking. **Phase 2** adds replayable historical-event episodes built from public products and published models, emphasizing realistic ambiguity. **Phase 3** introduces failure-heavy scenarios where candidate outputs conflict, preprocessing choices diverge, or forward validation reranks or rejects inversion outputs. These scenarios are crucial for evaluating robust scientific decision-making rather than simple solver invocation (Mai et al., 2016a).

Each episode follows a reproducible pipeline: assemble event and observations; preprocess waveforms and metadata with ObsPy, Pyrocko, and optional gmprocess (Krischer et al., 2015; Heimann et al., 2017; International Federation of Digital Seismograph Networks (FDSN), 2019; 2022; 2023); generate or ingest source candidates with Grond; run finite-fault inversion with WISP (Heimann et al., 2018; Goldberg et al., 2026); validate selected candidates with OpenSWPC, SeisSol, SPECFEM, or SW4 (Maeda et al., 2017); optionally attach a GeoGPT[1]-based supplemental review over inversion/validation artifacts and disagreement evidence; and store artifacts, manifests, traces, failure labels, evaluator artifacts, and resource logs under a common schema. Input modalities include strong-motion data (near-source shaking records), teleseismic data (distant earthquake recordings), GNSS (ground-displacement measurements), and optional geodetic constraints. The initial benchmark release intentionally stops at rupture inversion plus forward validation; hazard, structural, and regional-impact stages are reserved for later extensions. Safety and privacy are handled through synthetic/public-data-first design, metadata redaction layers, and separation of shareable scientific state from restricted operational context.

[1]GeoGPT repository (`https://github.com/GeoGPT-Research-Project/GeoGPT`)

3. Metadata and Governance

Metadata and governance are core components, not release add-ons. The benchmark is designed for *comparable*, *reproducible*, and *responsibly shared* evaluation, consistent with finite-fault standardization and validation lessons (Mai et al., 2016b;a). Each episode records event and waveform metadata, preprocessing settings, velocity-model references, inversion requests, candidate and validation outputs, traces, budget metadata, split assignment, preprocessing-version identifiers, and environment fingerprints.

A shared object layer (`EventHypothesis`, `WaveformBundle`, `VelocityModelRef`, `InversionRequest`, `CandidateResult`) acts as the contract between agents and backend adapters, preserving scientific meaning across tools. It enables automatic checks of schema conformance, field completeness, and preprocessing consistency before outputs enter comparative scoring. Labels are grouped as reference, comparative, consistency, and workflow.

Governance combines clear licensing with tiered access. When redistribution is constrained, reconstruction manifests (identifiers, hashes, transforms) are provided. Open, controlled, and redacted modes balance usability with sensitivity while keeping split and provenance requirements consistent.

4. Acceleration Potential

The resource targets milestones beyond single-tool success: *inversion-to-validation orchestration* (when to branch from candidate generation into finite-fault inversion and physics-based checking), *failure-aware reasoning* (detecting disagreement, reranking candidates, or abstaining rather than trusting the first solver), and *reproducible execution* (maintaining preprocessing consistency, provenance, and budget-aware behavior across workflows) (Yao et al., 2022; Wang et al., 2023; Schick et al., 2023; Chen et al., 2024).

This benchmark also supports uncertainty-aware behavior. Because rupture solutions are often non-unique, high-quality agents should treat disagreement as a signal for reranking, abstention, validation, or escalation (Mosegaard & Tarantola, 1995; Minson et al., 2013). The benchmark serves as a practical testbed for AI systems operating in constrained scientific software ecosystems.

5. Conclusion

We presented SeisAgentBench as a failure-aware core benchmark for evaluating multi-agent earthquake rupture workflows through standardized, replayable episodes with shared schema objects and provenance-aware traces.

By focusing the initial release on source exploration, finite-fault inversion, and wave-simulation validation, the resource supports narrow but realistic and auditable evaluation of AI-assisted seismology workflows.

References

Chen, Z., Chen, S., Ning, Y., Zhang, Q., Wang, B., Yu, B., Li, Y., Liao, Z., Wei, C., Lu, Z., et al. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*, 2024.

Goldberg, D. E., Hunsinger, H. E., Koch, P., Haynie, K. L., Melgar, D., and Riquelme, S. Wavelet inversion for slip (wisp): Open-source earthquake slip modeling software. *Seismological Research Letters*, 2026. doi: 10.1785/0220250055. URL https://doi.org/10.1785/0220250055. metadata incomplete.

Heimann, S., Kriegerowski, M., Isken, M., Cesca, S., Daout, S., Grigoli, F., Juretzek, C., Megies, T., Nooshiri, N., Steinberg, A., Sudhaus, H., Vasyura-Bathke, H., Willey, T., and Dahm, T. Pyrocko - an open-source seismology toolbox and library, 2017. URL https://doi.org/10.5880/GFZ.2.1.2017.001.

Heimann, S., Isken, M., Kühn, D., Sudhaus, H., Steinberg, A., Daout, S., Cesca, S., Vasyura-Bathke, H., and Dahm, T. Grond - a probabilistic earthquake source inversion framework, 2018. URL https://doi.org/10.5880/GFZ.2.1.2018.003.

International Federation of Digital Seismograph Networks (FDSN). Fdsn web service specification: fdsnws-dataselect (version 1.1, 2019-06-27). Specification, 2019. URL https://doi.org/10.7914/QFEP-3Z75.

International Federation of Digital Seismograph Networks (FDSN). Stationxml documentation (version 1.2, 2022-02-25). Online documentation, 2022. URL https://docs.fdsn.org/projects/stationxml/en/v1.2/.

International Federation of Digital Seismograph Networks (FDSN). miniseed 3: Specification (rev. 2023-01-18). Online documentation, 2023. URL https://docs.fdsn.org/projects/miniseed3/en/latest/.

Krischer, L., Megies, T., Barsch, R., Beyreuther, M., Lecocq, T., Caudron, C., and Wassermann, J. Obspy: a bridge for seismology into the scientific python ecosystem. *Computational Science & Discovery*, 8(1):014003, 2015. doi: 10.1088/1749-4699/8/1/014003. URL https://doi.org/10.1088/1749-4699/8/1/014003.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H.,

Huang, M., Dong, Y., and Tang, J. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Maeda, T., Takemura, S., and Furumura, T. Openswpc: An open-source integrated parallel simulation code for modeling seismic wave propagation in 3d heterogeneous viscoelastic media. *Earth, Planets and Space*, 69:102, 2017. doi: 10.1186/s40623-017-0687-2. URL https://doi.org/10.1186/s40623-017-0687-2.

Mai, P. M., Schorlemmer, D., Page, M. T., Ampuero, J.-P., Asano, K., Causse, M., Cotton, F., Custódio, S., Fan, W., Festa, G., Gallovič, F., Gholami, V., Graves, R. W., Hellweg, M., Kaneko, Y., Kiremidjian, A., Lowry, A., Ma, S., Mena, B., Minson, S. E., Nihira, K., O'Toole, T. B., Polet, J., Putra, R. R., Shahraeeni, S., Shimizu, K., Somala, S. N., Suzuki, W., Tsuda, K., Vassiliou, M. S., Wimpenny, S., Zahradník, J., and Zielke, O. The earthquake-source inversion validation (siv) project. *Seismological Research Letters*, 87(3):690–708, 2016a. doi: 10.1785/0220150231. URL https://doi.org/10.1785/0220150231.

Mai, P. M., Shearer, P. M., Ampuero, J.-P., and Lay, T. Standards for documenting finite-fault earthquake rupture models. *Seismological Research Letters*, 87(3):712–718, 2016b. doi: 10.1785/0220150204. URL https://doi.org/10.1785/0220150204.

Minson, S. E., Simons, M., and Beck, J. L. Bayesian inversion for finite fault earthquake source models i: Theory and algorithm. *Geophysical Journal International*, 194 (3):1701–1726, 2013. doi: 10.1093/gji/ggt180. URL https://doi.org/10.1093/gji/ggt180.

Mosegaard, K. and Tarantola, A. Monte carlo sampling of solutions to inverse problems. *Journal of Geophysical Research: Solid Earth*, 100(B7):12431–12447, 1995. doi: 10.1029/94JB03097. URL https://doi.org/10.1029/94JB03097.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Starace, G., Jaffe, O., Sherburn, D., Aung, J., Chan, J. S., Maksin, L., Dias, R., Mays, E., Kinsella, B., Thompson, W., et al. Paperbench: Evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*, 2023.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

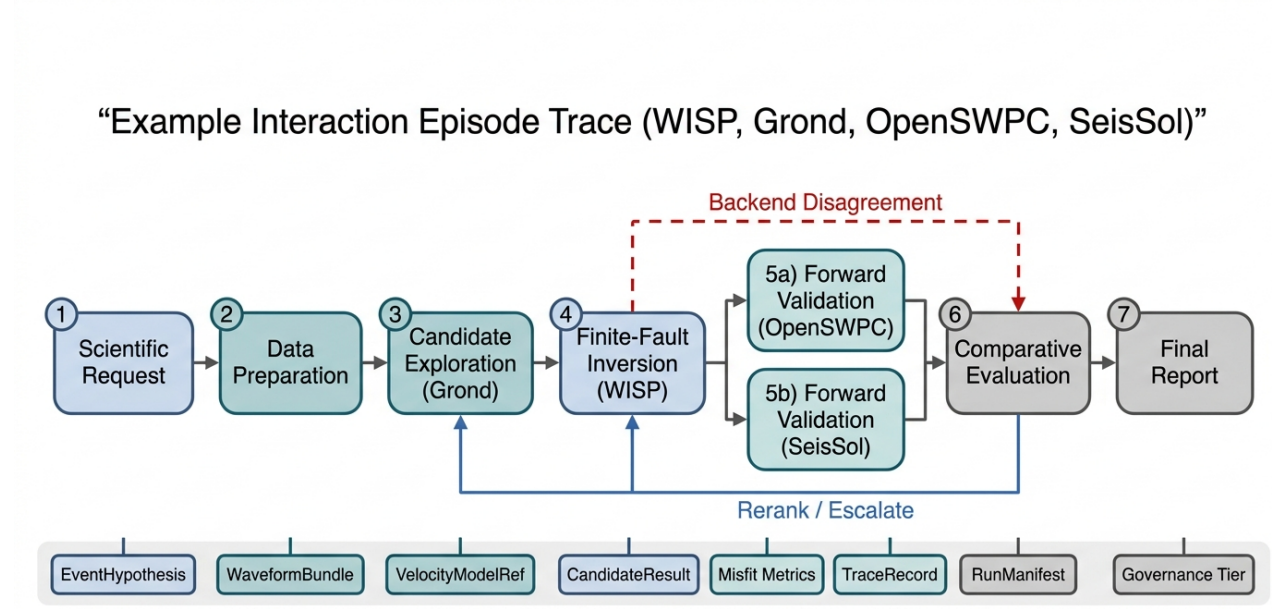

Figure 2. An interaction-episode execution trace. The final version will visualize the timeline of scientific request handling, preprocessing, Grond-based candidate exploration, WISP inversion, OpenSWPC/SeisSol/SPECFEM/SW4 forward validation, disagreement-aware reranking or abstention, failure labeling, and provenance-rich final reporting.

A. Appendix Overview

This appendix provides the technical details omitted from the main text for space reasons. It is organized into four parts. Appendix B describes the multi-agent method and backend orchestration design. Appendix C summarizes the core OSS coverage and the role of each software family. Appendix D specifies the shared schema used to standardize interaction episodes across heterogeneous scientific software. Appendix E details the evaluation protocol, metrics, baseline settings, and failure-aware validation logic used to assess the usefulness of the proposed resource for AI-for-Science systems. An illustrative execution-trace figure is included in Fig. 2.

B. Method Details

B.1. Method Summary

The proposed resource, SeisAgentBench, is designed as a failure-aware core benchmark around a multi-agent computational workflow for earthquake rupture-process analysis. The central motivation is that realistic rupture-analysis work rarely consists of a single model invocation. Instead, researchers move through several stages: hypothesis formation, data conditioning, inversion, forward validation, and cross-model comparison. Each of these stages is often handled by different software systems, and each introduces different assumptions about geometry, waveform preprocessing, uncertainty, and computational cost.

To support AI systems that operate over such workflows, we define the benchmark unit as an interaction episode involving structured scientific state and tool-mediated execution. Each episode records the objects and decisions needed to reconstruct how a candidate rupture model was proposed, how it was evaluated, and how competing models were compared. The purpose of the method layer is not to replace physics-based solvers with language models, but to expose the orchestration problem itself as a first-class target for AI-for-Science research.

Figure 2 provides a compact, end-to-end visualization of one interaction episode, including candidate generation, inversion, multi-backend validation, failure labeling, and governance-aware reporting.

B.2. Multi-Agent Architecture

The reference architecture contains layered roles across the core workflow. A **Supervisor Agent** interprets the scientific request and routes execution across preprocessing, source exploration, inversion, and forward-validation stages. A **Data Agent** and **Preprocessing Agent** normalize waveform, station, and ground-motion products through ObsPy, Pyrocko, and optional gmprocess. Source-model stages are handled by **Grond** and **WISP** agents. Wave-propagation validation is handled by **OpenSWPC**, **SeisSol**, **SPECFEM**, and **SW4** agents. An optional **GeoGPT-based Review Module** may summarize disagreement evidence after primary scoring, but its outputs are stored only as supplemental evaluation artifacts.

These roles need not be implemented as separate large language model instances. In practice, they may be instantiated as a mixture of typed software components, backend adapters, rule-based controllers, and language-mediated planning modules. The key design choice is modularity: no backend-specific logic should leak into the shared representation layer, and no cross-backend comparison should be performed on raw backend-native output alone.

B.3. Scientific Workflow as an Episode

We represent an episode as a sequence of scientifically meaningful transitions in which a user or benchmark specification first provides an event-centered objective, the system constructs a normalized event description and observation bundle, one or more rupture hypotheses are generated, a finite-fault inversion backend refines candidate parameters, a forward-simulation backend evaluates selected candidates under explicit structural assumptions, candidate outputs are compared under harmonized metrics, and all artifacts, traces, and decisions are stored in a reproducible record.

This formulation makes it possible to study AI behavior at several levels: task interpretation, workflow planning, backend selection, cross-backend comparison, provenance management, and validation-aware scientific decision making.

B.4. Backend Roles

B.4.1. Grond for Candidate Exploration

Grond is used as an exploratory backend when the workflow requires probabilistic source-model search or broad candidate generation before detailed rupture inversion. In the benchmark setting, Grond plays two roles. First, it provides initial source hypotheses when event geometry is uncertain. Second, it provides alternative candidate representations against which more detailed finite-fault inversions can be compared.

Because Grond can be used under different problem setups, the benchmark does not assume a single canonical Grond task type. Instead, Grond-derived outputs are normalized into the shared representation at the level of event hypothesis, candidate metadata, and evaluation artifacts. This abstraction is important for preserving comparability even when backend-native configuration spaces differ across experiments.

B.4.2. WISP for Finite-Fault Inversion

WISP serves as the main rupture-process inversion backend in the reference workflow. It is used for tasks in which slip amplitude, rupture timing, rake, and related finite-fault parameters must be estimated from observational data. In the benchmark, WISP-generated candidates are stored not only as final model artifacts but as part of a richer episode object that includes input assumptions, preprocessing context, and post-inversion evaluation.

By wrapping WISP in a shared interface, the resource supports questions that go beyond "Did the inversion converge?" For example, one can ask whether the system chose WISP appropriately, whether its inputs were scientifically consistent, and whether the resulting model remained competitive after external forward validation.

B.4.3. Wave-Propagation Backends for Forward Validation

OpenSWPC, SeisSol, SPECFEM, and SW4 are used in two ways. First, they generate synthetic and semi-synthetic data for controlled episodes in which source and medium conditions are known. Second, they serve as validation backends that re-simulate selected candidate rupture models under explicit structural assumptions. This second role is central to the benchmark design because it prevents the workflow from treating a backend-native objective value as the only notion of success.

A candidate model that fits an inversion objective well but fails under forward simulation is not merely a failed example; it is

scientifically informative. The benchmark therefore treats forward-validation disagreement as a labeled event type that can be used for evaluation, training, and analysis.

B.4.4. GeoGPT as a Supplemental Review Artifact

GeoGPT (`https://github.com/GeoGPT-Research-Project/GeoGPT`) is not part of the scientific ground truth or the primary score computation. When enabled, it reviews inversion and forward-validation artifacts after core metrics are computed, then emits rubric-aligned notes about consistency, plausibility, and unresolved conflict.

To keep review behavior auditable, GeoGPT outputs are stored as optional supplemental artifacts with metadata such as model/version identifiers, prompt-template identifiers, and input-context hashes.

B.5. Routing Logic

The benchmark is designed to support structured tool choice rather than unconstrained tool calling. A reference routing strategy is to begin with ObsPy/Pyrocko/gmprocess preprocessing, use Grond for broad candidate search when geometry is uncertain, run WISP for finite-fault inversion, run one or more forward backends (OpenSWPC/SeisSol/SPECFEM/SW4) for structural validation, compute comparative scores and failure labels, and optionally attach a GeoGPT supplemental review artifact. Major backend disagreement is stored explicitly and escalated to comparative evaluation or abstention rather than collapsed to a single answer.

This routing policy is intentionally simple and interpretable. The goal of the resource is not to prescribe one universal orchestration policy, but to provide a shared substrate on which such policies can be trained and evaluated.

B.6. Human Oversight and Safety Gates

The reference workflow includes approval gates for expensive or high-impact actions, including launching large 3D forward simulations, changing fault geometry after partial inversion, promoting one candidate as the canonical episode result, and executing pipelines that depend on non-shareable or restricted data.

Human oversight is included not because the benchmark assumes weak automation, but because realistic scientific AI systems must be evaluated under the same governance and caution that apply to human-run computational science workflows.

B.7. Illustrative End-to-End Workflow

A representative episode may proceed as follows. A user requests a rupture-analysis benchmark evaluation for a historical event with strong-motion and GNSS data. The Supervisor Agent routes preprocessing through ObsPy/Pyrocko and gmprocess, then sends the workflow to Grond for initial candidate generation and to WISP for detailed finite-fault inversion. Selected candidates are transformed into source descriptions for OpenSWPC, SeisSol, SPECFEM, or SW4, and the resulting validation products are compared under harmonized metrics. Contradictory evidence triggers reranking, abstention, or an unresolved outcome rather than silent acceptance. If enabled, a GeoGPT-based Review Module summarizes disagreement evidence and stores a supplemental review artifact in the episode object.

C. OSS Integration Coverage

This section summarizes the role of each open-source software family in the core SeisAgentBench release and notes adjacent tools that motivate future extensions. URLs are provided to clarify software scope and expected integration points.

C.1. Grond (Source Candidate Exploration)

Grond (`https://pyrocko.org/grond/`) is used for probabilistic source-model exploration and candidate generation when geometry and mechanism assumptions are uncertain. In SeisAgentBench episodes, Grond provides broad candidate sets and uncertainty-aware summaries that are later refined by inversion and validation stages.

C.2. WISP (Finite-Fault Inversion)

WISP is used as the detailed finite-fault inversion backend that refines candidate rupture parameters from waveform and geodetic constraints. Within the benchmark workflow, WISP bridges exploratory source hypotheses and downstream

forward-validation stages by producing structured rupture outputs suitable for cross-backend comparison.

C.3. OpenSWPC (3D Wave Propagation Validation)

OpenSWPC (`https://openswpc.github.io/`) is used as a reference 3D wave-propagation backend for physics-based validation of rupture candidates under explicit structural assumptions. Its outputs are evaluated jointly with other simulators to determine whether candidate reranking or disagreement escalation is required.

C.4. GeoGPT (Supplemental Review Artifact)

GeoGPT (`https://github.com/GeoGPT-Research-Project/GeoGPT`) is integrated only as an optional geoscience-domain review module. In SeisAgentBench episodes, it does not replace deterministic solvers or contribute primary scores; instead, it consumes inversion/validation outputs and disagreement traces, then emits rubric-based reports for qualitative analysis.

C.5. Future Extensions Beyond v1

Hazard, structural-response, and regional-impact software such as OpenSHA, OpenQuake, EE-UQ, and R2D motivate broader workflow research, but they are explicitly out of scope for the initial SeisAgentBench release. The core benchmark is intentionally narrower so that evaluation can focus on rupture inversion, forward validation, and failure handling before expanding downstream.

C.6. gmprocess, ObsPy, and Pyrocko (Preprocessing and Analysis)

gmprocess, ObsPy (`https://docs.obspy.org/`), and Pyrocko (`https://pyrocko.org/`) support waveform retrieval, conditioning, metadata harmonization, and event-level preprocessing. In SeisAgentBench episodes, these tools anchor the preprocessing layer so that inversion and forward-validation comparisons are made under controlled assumptions.

C.7. SPECFEM, SeisSol, and SW4 (Additional High-Fidelity Simulation Backends)

SPECFEM (`https://specfem.org/`), SeisSol (`https://seissol.org/`), and SW4 (`https://github.com/geodynamics/sw4`) provide additional high-fidelity wave propagation and dynamic rupture simulation capabilities. They complement OpenSWPC by broadening the forward-simulation envelope and enabling disagreement-aware validation across multiple physics engines.

D. Schema Specification

D.1. Design Goals

The schema is designed to satisfy four requirements: **scientific comparability**, so results from different software systems remain comparable at the level of scientific meaning; **reproducibility**, so episodes store enough information to reconstruct the workflow; **extensibility**, so new data types, backends, and validation stages can be incorporated without breaking the representation; and **responsible sharing**, so sharable and non-sharable content remain separable.

To satisfy these requirements, we separate *scientific objects* from *backend artifacts*. Scientific objects capture normalized meaning, whereas backend artifacts capture implementation-specific traces and files.

D.2. Top-Level Episode Object

Each benchmark item is a top-level `Episode` object whose conceptual fields are `episode_id` (unique identifier), `task_family` (synthetic generation, historical replay, cross-backend comparison, or recovery-oriented validation), `objective`, `event`, `waveforms`, `velocity_model`, `requests`, `candidates`, `comparisons`, `failure_labels`, `evaluator_reports` (optional supplemental review artifacts), `traces`, `budget`, `governance`, and `split` (train, validation, test, held-out, or challenge).

D.3. Core Scientific Objects

D.3.1. EventHypothesis

`EventHypothesis` represents a candidate scientific description of an earthquake source. Required fields are `event_id`, `origin_time`, `latitude`, `longitude`, `depth_km`, `mechanism` (strike, dip, rake), `fault_geometry` (type, dimensions, discretization), and `source_reference`, with `magnitude_mw` treated as optional.

The key rule is that `EventHypothesis` encodes the scientific interpretation, not the backend-specific file layout.

D.3.2. WaveformBundle

`WaveformBundle` represents the observational input used by the episode and includes `bundle_id`, `waveform_dir` (or external references), `station_metadata_path`, `formats`, `components`, `data_types`, `preprocess_recipe_id`, and `time_reference`.

A preprocessing recipe is treated as a first-class object because cross-backend comparison is invalid if waveform assumptions are inconsistent.

D.3.3. VelocityModelRef

`VelocityModelRef` describes the structural model used in generation or validation through fields such as `model_id`, `source`, `path` (or a reconstruction recipe), `bounds`, `resolution`, `converter`, and `notes`.

The schema distinguishes between a model reference and a backend-specific meshed or discretized representation. This is necessary because a single structural model may be projected differently for different solvers.

D.3.4. InversionRequest

`InversionRequest` defines the scientific and computational intent via `request_id`, `preferred_backend`, `objective`, `estimate` flags (e.g., slip, rake, rupture time, rise time), `constraints`, `regularization`, `validation` settings, and `approval_required`.

This object ensures that the benchmark stores not only what was done, but what was intended.

D.3.5. CandidateResult

`CandidateResult` stores normalized outputs from a backend run, including `candidate_id`, `backend`, `event_hypothesis_id`, artifact paths, `metrics`, `status`, and `notes`.

Candidates may come from inversion, validation, or comparative reranking. The schema therefore allows multiple candidate layers within one episode.

D.4. Traces and Provenance

To make episodes reconstructable, we store action- and execution-level provenance. A `TraceRecord` includes timestamp, actor or component identifier, action type, input and output object identifiers, environment summary, success or failure status, and a free-text note or machine-readable warning. In addition, each episode includes a `RunManifest` that records software versions, container hashes (if applicable), configuration hashes, data hashes where possible, command summaries, and resource-usage summaries.

This provenance layer is required for responsible reuse. Without it, the benchmark would capture only products, not process.

D.5. Evaluator Reports

Each episode may store one or more `EvaluatorReport` objects produced by an optional GeoGPT-based review module. Recommended fields are `evaluator_id`, `model_version`, `prompt_template_id`, `input_context_ids`, `rubric_scores`, `verdict`, and `rationale`.

These reports are treated as supplemental evaluation artifacts rather than ground truth labels or primary scoring inputs, and are designed to be auditable and replayable under fixed evaluator settings.

D.6. Labeling Standards

Because the resource includes both synthetic and real-event settings, labels are typed rather than monolithic. *Reference labels* are used in synthetic and semi-synthetic settings with known source parameters or construction targets. *Comparative labels* are used when multiple candidates must be ranked or backend disagreement must be preserved rather than collapsed. *Consistency labels* indicate whether forward validation supports, weakly supports, or contradicts an inversion candidate. *Failure labels* record benchmark-relevant breakdowns such as incompatible preprocessing, missing metadata, forward-validation contradiction, unresolved disagreement, or budget exhaustion. *Workflow labels* capture non-scientific but essential conditions such as schema completeness, preprocessing consistency, and provenance completeness. *Governance labels* indicate whether an episode is openly shareable, reconstruction-only, partially redacted, or subject to restricted metadata handling.

This labeling system is intentionally layered because real-event rupture inversion often does not admit a single final-answer label.

D.7. Licensing and Access Fields

Each episode stores explicit governance fields. The core license and access keys are `license_metadata`, `license_waveforms`, `access_tier`, and `redistribution_allowed`; additional traceability keys are `license_derived_artifacts`, `reconstruction_recipe_available`, and `sensitivity_flags`.

These fields allow downstream users to filter for fully redistributable episodes or to reconstruct benchmark content from upstream sources when direct redistribution is not allowed.

D.8. Example Episode Skeleton

An illustrative skeleton for one episode is:

```
Episode
|-- episode_id
|-- task_family
|-- objective
|-- event: EventHypothesis
|-- waveforms: WaveformBundle
|-- velocity_model: VelocityModelRef
|-- requests:
|   |-- inversion_request_1
|   `-- validation_request_1
|-- candidates:
|   |-- candidate_grond_1
|   |-- candidate_wisp_1
|   `-- candidate_validation_1
|-- comparisons:
|   `-- evaluation_report_1
|-- failure_labels:
|   `-- failure_label_1
|-- evaluator_reports:
|   `-- geogpt_report_1
|-- traces:
|   |-- trace_record_1
|   |-- trace_record_2
|   `-- ...
|-- budget
|-- governance
`-- split
```

E. Evaluation Protocol

E.1. Evaluation Philosophy

The purpose of evaluation is not only to test whether an agent can produce a rupture model, but whether it can operate reliably inside a realistic scientific workflow. Accordingly, the benchmark evaluates three complementary dimensions: **scientific adequacy** (whether resulting candidates remain scientifically plausible), **workflow competence** (whether required orchestration steps are executed coherently and reproducibly), and **failure-aware reasoning** (whether multiple backends are used appropriately and disagreement is interpreted responsibly).

This design reflects the fact that a useful AI system for rupture analysis must do more than optimize one local objective.

GeoGPT, when enabled, is used in this protocol only to produce supplemental rubric-aligned review artifacts from inversion/validation evidence and disagreement traces, while physics-based adequacy metrics remain separately computed.

E.2. Task Families for Evaluation

We define four broad evaluation families.

E.2.1. Synthetic Generation Episodes

These episodes test whether a system can recover or compare source models when data are generated under controlled conditions. They are useful for early-stage benchmarking because the source construction is known and variations in geometry, structure, and observation subsets can be manipulated systematically.

E.2.2. Historical Replay Episodes

These episodes test performance on public or reconstructable real-event workflows. The goal is not to recover one official truth, but to evaluate how systems operate when multiple plausible models exist and published solutions disagree.

E.2.3. Cross-Backend Comparison Episodes

These episodes evaluate whether systems can reason across heterogeneous backends. Typical tasks include deciding when to run OpenSWPC or SeisSol validation after WISP inversion, or identifying when Grond and WISP disagree strongly enough to require explicit comparison rather than silent model selection.

E.2.4. Recovery and Consistency Episodes

These episodes focus on robustness. They may include mismatched preprocessing, incomplete structural coverage, incompatible geometry definitions, or forward-validation contradiction. The system is evaluated on whether it flags these issues, routes work appropriately, and preserves provenance.

E.3. Scientific Adequacy Metrics

Scientific adequacy metrics are computed at the candidate level and, where possible, under harmonized observation bundles. They include normalized waveform misfit (the discrepancy between observed and simulated waveforms after standardized preprocessing), variance reduction relative to a baseline or reference fit, moment consistency with expected source-moment ranges or known event-scale summaries, slip roughness or smoothness penalties as structural regularity measures, rupture-velocity plausibility checks for physically implausible propagation patterns, cross-modal consistency checks that prevent gains in one data modality from being accepted when they cause unacceptable degradation in another, and validation residual consistency across forward simulators when more than one physics engine is used.

These metrics are not intended to declare a unique truth for real events. Instead, they provide a consistent basis for candidate comparison.

E.4. Workflow Competence Metrics

Workflow competence measures how well a system navigates the episode independent of any one candidate score. Representative metrics include schema validity rate (the fraction of steps producing schema-conformant objects), preprocessing

consistency rate across candidate comparisons, provenance completeness for required hashes, manifests, traces, and environment summaries, optional evaluator-artifact completeness for required GeoGPT metadata fields, backend-routing appropriateness relative to objectives and available data, reproducibility score based on whether another user can reconstruct the episode from released metadata and recipes, and budget compliance within declared runtime and resource envelopes.

E.5. Failure-Aware Reasoning Metrics

These metrics specifically target multi-agent intelligence over heterogeneous tools by checking whether the system detects material disagreement between backends in geometry, timing, or validation behavior; whether it handles disagreement by escalating to comparative analysis or abstention instead of silently selecting one output; whether it reranks candidates after forward validation (OpenSWPC/SeisSol/SPECFEM/SW4); whether recorded failure labels match the observed evidence; and whether optional GeoGPT review artifacts remain consistent with the same evidence.

E.6. Reference Baselines

The benchmark is designed to support multiple baseline families, including single-backend baselines (e.g., Grond-only, WISP-only, or inversion without forward validation), rule-based pipeline baselines (e.g., Grond $\rightarrow$ WISP $\rightarrow$ OpenSWPC/SeisSol/SPECFEM/SW4), agentic baselines that select tools dynamically while operating over the shared schema, evaluator-enabled baselines that include optional GeoGPT review artifacts, evaluator-ablated baselines without GeoGPT review, and oracle-assisted baselines in which selected fields are pre-normalized to isolate later-stage orchestration ability.

These baseline families are useful because they separate the gains due to better scientific modeling from gains due to better orchestration.

E.7. Split Design

The benchmark supports multiple split types: a **train split** for development episodes, a **validation split** for model selection and ablation, a **test split** of held-out episodes with no public labels beyond governance metadata, and a **challenge split** emphasizing cross-backend disagreement, explicit failure modes, incomplete information, or computational constraints.

In addition, we recommend at least one *held-out scenario split* in which event types, structural settings, or observation combinations differ from those seen during development. This prevents systems from overfitting to one narrow orchestration pattern.

E.8. Evaluation Procedure

A standard evaluation run loads an episode and validates access permissions, reconstructs all shareable scientific objects, executes the system under the declared budget, stores generated source and validation objects together with traces and manifests, optionally invokes the GeoGPT review module on consolidated episode evidence, stores evaluator reports when enabled, computes scientific adequacy metrics and workflow competence metrics, computes failure-aware reasoning metrics when more than one backend stage is present, and finally produces a report with candidate rankings, failure categorizations, optional review judgments, and governance-aware artifacts.

To maintain fairness, the evaluation harness should reject candidate comparisons that violate preprocessing consistency or use unavailable data tiers.

E.9. Failure and Warning Categories

Because SeisAgentBench is intentionally failure-aware, a structured warning system is part of the evaluation core. Recommended warning categories include instruction ambiguity, incomplete waveform metadata, structural-model coverage mismatch, backend configuration incompatibility, forward-validation contradiction, unresolved inversion/validation disagreement, evaluator-context incompleteness in optional GeoGPT stages, provenance incompleteness, budget overrun, and governance-violation risk.

These categories help distinguish scientific disagreement from operational breakdown.

E.10. Success Criteria

An episode is considered successfully completed when the system either produces at least one valid candidate result or explicitly abstains/escalates for the correct failure-aware reason, the outcome is accompanied by complete required provenance, all comparisons are performed under consistent preprocessing assumptions, governance restrictions are respected, and the final report explicitly indicates whether the outcome is provisional, validated, or unresolved under cross-backend disagreement. When GeoGPT review is enabled, its artifacts must include the required metadata.

This definition is intentionally stronger than "a solver produced an output." It aligns success with scientific usability.

F. Limitations and Scope

The proposed resource is intentionally designed to support meaningful progress without assuming that rupture-process inversion admits a unique answer in all settings. Several limitations and scope boundaries should therefore be stated explicitly.

First, real-event episodes often do not provide ground truth in the same sense as synthetic episodes. The benchmark addresses this by using layered labels and comparative evaluation rather than forcing a single-answer view of rupture analysis. Second, forward validation depends on structural assumptions, meaning that disagreement between inversion and simulation may arise from source-model error, structure-model error, or both. Third, the initial SeisAgentBench release is deliberately narrower than a full rupture-to-risk workflow: hazard, structural-response, and regional-impact stages are future extensions rather than current benchmark commitments. Fourth, licensing and access constraints may differ across waveform products and derived artifacts, requiring governance-aware release mechanisms. Finally, high-fidelity 3D validation can be computationally expensive, so not all benchmark users will be able to execute the most demanding episode variants.

These limitations are not weaknesses of the benchmark alone; they reflect the scientific reality of rupture-process analysis. The benchmark is intended to make such constraints explicit and measurable, not to pretend they do not exist.

G. Appendix Conclusion

This appendix described the technical details of the proposed resource at the level of method, OSS integration, schema, and evaluation. The central design principle is that AI-for-Science progress in earthquake rupture workflows depends not only on stronger inversion algorithms, but also on structured representations and reproducible orchestration across preprocessing, inversion, and forward-validation software stacks. By organizing the resource around interaction episodes, shared scientific objects, optional GeoGPT review artifacts, explicit failure labels, and explicit evaluation across Grond, WISP, OpenSWPC, SeisSol, SPECFEM, and SW4, the benchmark provides a concrete substrate for studying how future scientific agents can reason, validate, and fail more responsibly in computational geophysics.