The Journal of Open Source Software

ado: a Python framework for computational experimentation and benchmarking

Michael A. Johnston [1*¶] **and Alessandro Pomponio** [1*]

1 IBM Research - Ireland ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.10304

Software

- Review
- Repository
- Archive

Editor: Matt Graham

Reviewers:

- @gmarupilla
- @mzrghorbani

Submitted: 13 March 2026
Published: 22 May 2026

Summary

The **Accelerated Discovery Orchestrator (ado)** is a Python package that addresses a recurring challenge in research software development: implementing common capabilities for design of experiments (DoE) and execution of related computational experiment campaigns. These cross-cutting capabilities span methodology (design-space specification, sampling, analysis), interface (CLI and configuration management), execution (parallel and scale-out), and data (sharing, provenance, and reuse).

ado delivers these capabilities across domains through a lightweight plugin model, where integrating new components can be as simple as decorating a Python function. This is enabled by ado's core abstraction: the *Discovery Space*.

Out-of-the-box, ado includes state-of-the-art optimization algorithms and predictive modeling tools, alongside experiments targeting foundation-model performance. Our aim is for ado to become a focal point for developing and consuming advanced capabilities for defining and executing experiment campaigns.

Statement of need

While the domains of computational science are diverse, spanning machine learning (ML), physics simulation, and hardware design, the process of experimentation is remarkably uniform. Whether tuning hyperparameters, benchmarking foundation models, or sweeping simulation parameters, researchers consistently follow a structured pattern: define a configuration space, select points, execute experiments, record results, and analyze the outcomes to guide the next iteration. This workflow, the *experiment campaign*, is central to modern research and development, yet it is often managed with tools that fail to capture its essential structure.

Scientific and ML workflow systems like Galaxy, AiiDA, and Kubeflow excel at executing general directed acyclic graphs (DAGs). However, they are fundamentally context-free, treating each step as a black box (George & Saha, 2022; Huber et al., 2020; The Galaxy Community, 2024). When it comes to experiment campaigns, this forces researchers to implement mechanisms for trial submission, parameter handling, and result collation.

ado directly addresses this gap. Instead of orchestrating arbitrary DAGs, ado focuses on the generic experiment campaign process. Users declaratively define configuration spaces and operations on them. ado then orchestrates the operations using its own protocols. For example, in sampling workflows, it handles reuse of prior measurements, trial execution and monitoring, and time-resolved measurement recording, maintaining consistency over a shared sample store. This approach mirrors the advantages of declarative systems like SQL or Terraform: reduced boilerplate, fewer errors, and greater clarity. It also aids code generators in creating experiment definitions and design spaces.

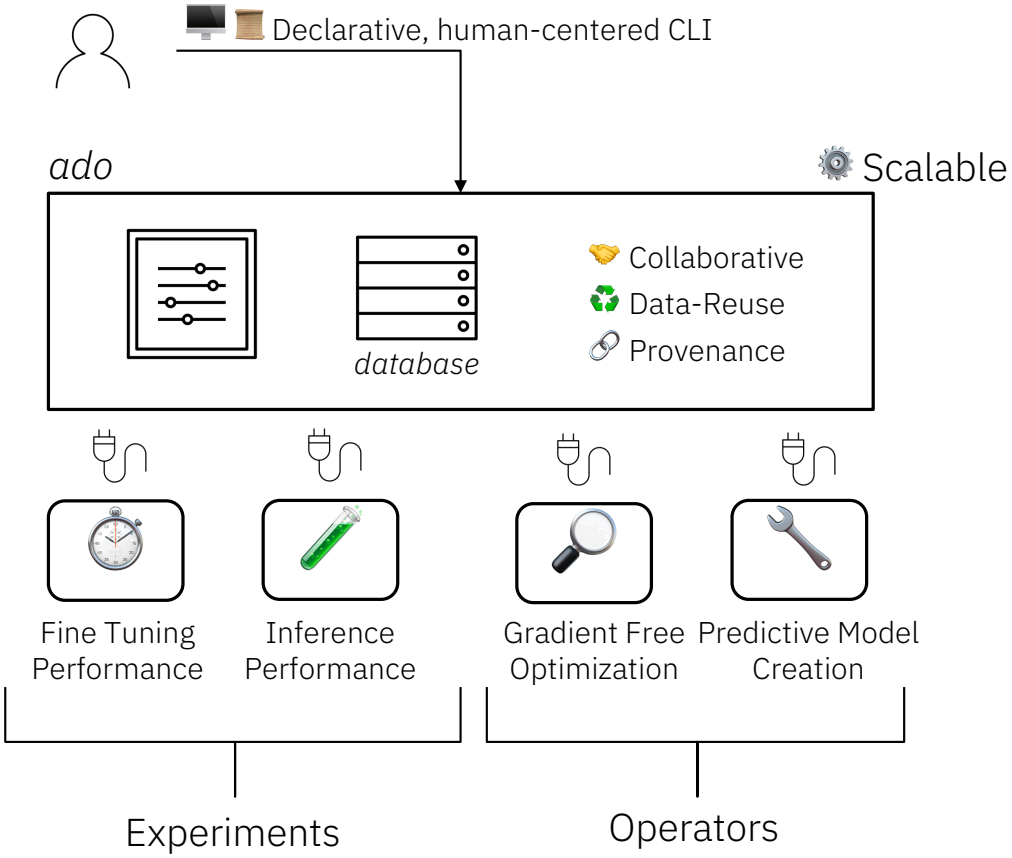

Figure 1: A schematic overview of ado's architecture.

ado extends its core model with valuable support capabilities. Specifications (configuration spaces, operations) and measurements are stored in a database with flexible deployment options (local or shared remote). It leverages Ray to seamlessly scale from a researcher's laptop to a large remote cluster (Moritz et al., 2018), with all functionality accessible via a human-centric CLI and Python API. Researchers can contribute custom experiments or operators through a simple plugin interface (see Figure 1).

State of the field

Mature workflow managers such as Galaxy, AiiDA, and Kubeflow excel at scalable, reliable DAG orchestration with strong provenance and native integration with common execution platforms (e.g., HPC for AiiDA; Kubernetes for Kubeflow). Their design prioritizes flexible, open-ended execution flow; adding the higher-level semantic constructs required for experiment campaigns would conflict with this architecture. Similarly, ML lifecycle management tools like MLflow provide robust experiment tracking, metric logging, and artifact management features for individual runs (Zaharia et al., 2018), but do not model the multi-stage lifecycle of a campaign as a primary entity.

Optimization frameworks like Optuna, Ax, Nevergrad, and Ray Tune are also key components for executing experiment campaigns (Akiba et al., 2019; Bennet et al., 2021; Liaw et al., 2018; Olson et al., 2025). While these tools are beginning to add data management features—for example, persistent storage for resuming studies—extending them to manage experiment campaigns would lead to duplicated effort. Furthermore, using these frameworks requires writing

glue logic between optimizers, objectives, and logging, which leads to bespoke implementations and reduced reusability.

Emerging robotic lab frameworks, for instance, the Experiment Orchestration System (EOS), provide rigorous, repeatable execution for physical experiments (Angelopoulos et al., 2025). The focus of these frameworks is managing the operational complexity of physical execution, answering how to carry out a specific repeatable experiment with lab instruments, rather than providing domain-agnostic campaign semantics.

ado synergizes with, rather than replaces, these tools. It fills a functional gap for experiment campaign management through a unified coordination layer that integrates cleanly with the existing ecosystem. It can use workflow managers as experiment executors, integrate optimization frameworks, and orchestrate physical experiments by coupling with robotic lab systems. At the same time, individual experiment implementations within ado's plugin architecture can leverage frameworks like MLflow for fine-grained, domain-specific tracking.

Software design

TRACE design requirements

We first established TRACE, a set of five requirements for managing the artifacts of an experimental campaign or study.

Characteristic	Description
Time-Resolved	The time series of sampling processes adding data to a study is preserved.
Reconcilable	There is a consistent protocol for adding data from a common context into a specific study.
Actionable	A study must contain all necessary information for adding a new measurement to itself.
Common Context	There is a schema and storage mechanism allowing data to be shared across multiple studies.
Encapsulated	The study rigorously defines the valid configurations and measurements, preventing contamination from unrelated data.

A system satisfying these requirements would ensure that a DoE and its associated data can be understood, shared, extended, and analyzed without introducing inconsistencies. In this way, the TRACE requirements offer a concrete implementation of the FAIR Principles (Findable, Accessible, Interoperable, and Reusable) (Wilkinson et al., 2016). Where FAIR describes *what* qualities a digital asset should possess, TRACE defines *how* to construct systems that generate inherently FAIR data from the inception of an experiment campaign.

Discovery Space as a core abstraction

The TRACE characteristics guided our search for a data model. First, we noted that configuration search campaigns have well-defined mathematical properties:

- **A Configuration Probability Space**: the definition of the dimensions of the configuration space being explored (the "what") and the probability distribution governing the selection process (the "how likely").
- **An Action Space:** the set of experiments that can be applied to a configuration to measure its properties (the "how to measure").
- **A sample set:** the set of points currently sampled and measured for a given combination of a configuration probability space and an action space. It is the union of the **sample time series** of operations on that combination.

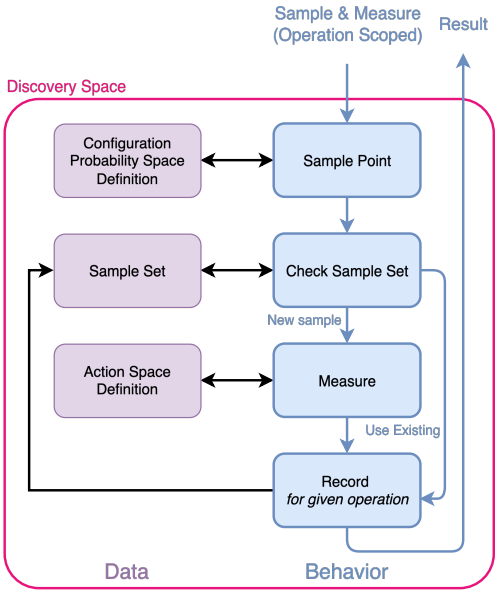

Figure 2: A view of a Discovery Space data model instance. The left-hand side shows the key data components of the model. The right-hand side shows the sampling and measurement process for a point.

The **Discovery Space**, the central data model in the system's architecture, combines these properties (see Figure 2). By containing the configuration probability space and action space definitions it is **encapsulated** and **actionable**; it is **time-resolved** as it contains the sample set.

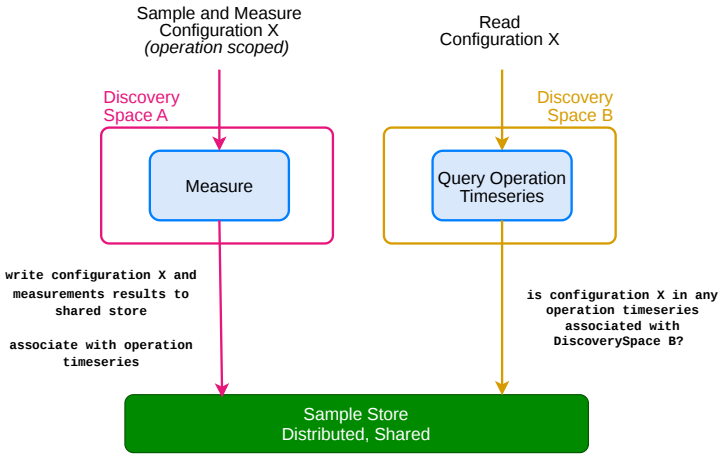

Figure 3: Data sharing between Discovery Spaces. Each space reads/writes samples and time series membership details from a shared sample store (*common context*). In this case both spaces have the same action space and contain point X. If point X is measured on Discovery Space A, it will not appear in the sample set of Discovery Space B until it is requested to be measured via B (*reconcilable*).

We obtain a **common context** by storing the sample time series in a shared sample store with a common schema. Finally, it is **reconcilable** as it enforces a strict membership rule for its sample set: only samples in the common-context that are associated with an operation conducted on that space are included (see Figure 3). Hence, it displays the TRACE characteristics.

The Discovery Space abstraction effectively decouples workload-specific experiments from

the search and optimization algorithms, enabling the kind of versatile, workload-agnostic capabilities that are a key goal of the program.

Pydantic-based core architecture

ado is implemented in Python, chosen for its ubiquity in scientific domains. Central to ado is the Pydantic framework, which we use for all data modeling. These Pydantic models serve as the primary user-facing components, providing automatic validation, self-documenting schemas, and a clear target for AI code generation.

They also serve as the data-contract for ado's extensible plugin architecture. This allows domain experts to contribute self-contained plugins for experiments, sampling, and analysis tools, which immediately and safely inherit all the platform's core capabilities

Research impact statement

ado is a community-ready platform for reproducible research, released as open source code with extensive documentation. It has been internally battle-tested on complex industrial workloads and research questions (Johnston et al., 2025). Its realized impact and utility are demonstrated by a range of publicly available artifacts.

- **Large-Scale Benchmarking:** All fine-tuning benchmarks for IBM's watsonx.ai platform were executed using ado. This effort produced the sft-trainer plugin, which we have open-sourced, a library of recommender models derived from the benchmark data, and a public dataset for benchmarking predictive models for fine-tuning performance (Lotito et al., 2026).
- **Advanced Performance Analysis:** ado was used to conduct detailed performance analysis of geospatial models on vLLM (Kwon et al., 2023). We have open-sourced the resulting vllm-performance plugin that includes unique features like automated deployment and tear-down of inference services on Kubernetes.
- **Accelerated Benchmarking:** We have open-sourced the TRIM operator plugin which we developed to reduce the time required to execute new benchmarking campaigns to account for availability of new workload options e.g. new GPUs, new model versions. It uses feature-importance-guided active learning to build performance surrogates with minimal measurements.

Based on this foundation we believe ado can positively impact research workflows in the near term. It lowers the barrier to complex benchmarking by eliminating the need to write bespoke orchestration code, while its plugin architecture provides a new avenue for researchers to distribute novel experiments, sampling methods, and analysis tools. Further, it enhances collaboration by giving research teams a shared environment for running experiments and storing data. We are actively developing ado to accelerate our own research and are frequently releasing new features, for example increasing its synergy with AI agents to enable more powerful automation of the experiment campaign lifecycle.

AI usage disclosure

Generative AI was used for the manuscript and codebase, with human authors reviewing and taking responsibility for the final content. For the manuscript, AI helped refine sentences and check for compliance with submission guidelines. For code, AI agents assisted with development tasks within a structured framework that includes coding rules, self-describing schemas with validation, and mandatory human review of all contributions.

Acknowledgements

ado is partially funded by the European Union through the Smart Networks and Services Joint Undertaking (SNS JU) under grant agreement No. 101192750 (Project 6G-DALI).

We acknowledge contributions from many people in the development of ado: Vassilis Vassiliadis, Christian Pinto, Srikumar Venugopal, Daniele Lotito, Michele Gazzetti, Burkhard Ringelin, Boris Lublinsky, Renato Maia, Renato Cerqueira, Gabriela Pinheiro, Raphael Melo, Christoph Hagleitner.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. https://doi.org/10.1145/3292500.3330701

Angelopoulos, A., Baykal, C., Kandel, J., Verber, M., Cahoon, J. F., & Alterovitz, R. (2025). The Experiment Orchestration System (EOS): Comprehensive foundation for laboratory automation. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 15900–15906. https://doi.org/10.1109/ICRA55743.2025.11128578

Bennet, P., Doerr, C., Moreau, A., Rapin, J., Teytaud, F., & Teytaud, O. (2021). Nevergrad: Black-box optimization platform. *SIGEVOlution*, *14*(1), 8–15. https://doi.org/10.1145/3460310.3460312

George, J., & Saha, A. (2022). End-to-end machine learning using kubeflow. *Proceedings of the 5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*. https://doi.org/10.1145/3493700.3493768

Huber, S. P., Zoupanos, S., Uhrin, M., Talirz, L., Kahle, L., Häuselmann, R., Gresch, D., Müller, T., Yakutovich, A. V., Andersen, C. W., Ramirez, F. F., Adorf, C. S., Gargiulo, F., Kumbhar, S., Passaro, E., Johnston, C., Merkys, A., Cepellotti, A., Mounet, N., … Pizzi, G. (2020). AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. *Scientific Data*, *7*(1), 300. https://doi.org/10.1038/s41597-020-00638-4

Johnston, M., Ringlein, B., Hagleitner, C., Pomponio, A., Vassiliadis, V., Pinto, C., & Venugopal, S. (2025). *Efficient and reusable cloud configuration search using discovery spaces*. https://doi.org/10.48550/arXiv.2506.21467

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626. https://doi.org/10.1145/3600006.3613165

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., & Stoica, I. (2018). Tune: A research platform for distributed model selection and training. *arXiv Preprint arXiv:1807.05118*. https://doi.org/10.48550/arXiv.1807.05118

Lotito, D., Venugopal, S., Vassiliadis, V., Pinto, C., Pomponio, A., & Johnston, M. (2026). *LLM fine-tuning performance benchmark dataset*. Hugging Face Datasets. https://huggingface.co/datasets/ibm-research/LLM_Fine-Tuning_Performance/

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & Stoica, I. (2018). Ray: A distributed framework for emerging AI applications. *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 561–577. https://doi.org/10.48550/arXiv.1712.05889

Olson, M., Santorella, E., Tiao, L. C., Cakmak, S., Eriksson, D., Garrard, M., Daulton,

S., Balandat, M., Bakshy, E., Kashtelyan, E., Lin, Z. J., Ament, S., Beckerman, B., Onofrey, E., Igusti, P., Lara, C., Letham, B., Cardoso, C., Shen, S. S., … Grange, M. (2025). Ax: A platform for adaptive experimentation. *AutoML 2025 ABCD Track*. https://openreview.net/forum?id=U1f6wHtG1g

The Galaxy Community. (2024). The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. *Nucleic Acids Research*, *52*(W1), W83–W94. https://doi.org/10.1093/nar/gkae410

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Silva Santos, L. B. da, Bourne, P. E., & others. (2016). The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, *3*. https://doi.org/10.1038/sdata.2016.18

Zaharia, M. A., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2018). Accelerating the machine learning lifecycle with MLflow. *IEEE Data Eng. Bull.*, *41*, 39–45. https://api.semanticscholar.org/CorpusID:83459546