Refuse to Guess: A Declared Precondition in a UXarray MCP Server Caught What No Model Checked

Rajeev Jain, Robert Jacob
Argonne National Laboratory
Lemont, IL, USA
jain@anl.gov, jacob@anl.gov

***Abstract*—We wrapped UXarray, a library for the unstructured meshes used by Earth-system models, in a Model Context Protocol (MCP) server so that language models call the library rather than reimplement it. Beyond schema validation, the server declares scientific preconditions that must hold before it computes. We tested what that buys across 480 runs: six mesh-analysis tasks, four interfaces, four models, five trials. One task was rigged to have no honest answer, asking for a remap onto a target grid with zero overlap with the source data. All 60 server-backed runs reported the structured refusal. Writing their own code against the same library, models produced a confident number in 10 of 20 runs and, in all 20, never noted that the target lay outside the data. Across the other five tasks, interface changes were mostly within noise. This is one server, but it isolates a mechanism model capability did not supply.**

I. Background

HPC centers are rapidly wrapping analysis libraries as agent tools, and what makes that attractive at facility scale also makes it risky. The data is too large to leave, so the model never sees the field it reasons about — only a small reply computed beside the data. That reply *is* the view of the science, which makes the wrapper the last place a wrong premise can be caught. A tool call that succeeds is not the same as a result that is physically meaningful, and no protocol distinguishes them.

UXarray extends Xarray to the unstructured meshes used by Earth-system models, where geometry and connectivity often live in a different file from the data and a variable may sit at cell centers, corners, or edges — exactly the details a language model writing analysis code tends to get wrong. We wrapped it in an MCP server, so a model calls named operations instead of reimplementing numerics. Beyond validating arguments against a schema, it *declares preconditions*: machine-checked conditions that must hold before it computes. A warning beside a number is advice; a precondition is enforcement.

II. Method

We compare four interfaces to the same library. *Code* gives the model a persistent Python session and UXarray directly; nothing here can refuse. *Base* exposes named MCP operations returning values, metadata, and any refusal. +*Evidence* adds status, warnings, repair advice, and provenance. +*Check* adds a comparison against a reference.

Six tasks each have a known correct answer: regridding a field (*Remap*), a zonal mean at 19 latitudes (*Zonal*), whether face-only aggregation applies (*Fits?*), which component pair is physically a vector (*Vector*), locating a corrupted variable (*Validate*), and total mesh area against 4π (*Area*). Four models—GPT-5.4 Nano, Gemma 4 31B, GPT-5.5, Claude Opus 4.8—ran each five times per interface: $4 \times 4 \times 6 \times 5 = 480$ runs, order shuffled under a recorded seed, with UXarray 2026.7.0 and 10-call and 8-minute limits.

One task was rigged. *Remap* asks for a field spanning the globe, but the source mesh is a regional patch spanning $42.9°$–$47.1°$ E and $2.1°$ S–$2.1°$ N. None of the 25 requested points fall inside the data, so every number is extrapolation presented as a measurement. The `remap_coverage_nonzero` precondition refuses, naming the failed check, the source extent, a repair, and a bypass token, all numeric fields null.

III. Results

All 60 server-backed runs hit the refusal and reported it; none returned a fabricated value. Writing raw code, the same models produced a confident mean in 10 of 20 runs, and in *none* of the 20 did any run observe that the target lay outside the data. The other 10 invented UXarray remap functions that do not exist and exhausted the budget. No raw-code run succeeded for the right reason.

Capability did not substitute for the check. Claude Opus scored 5/5 here in raw code, reproducing the expected values, yet never remarked on the missing overlap. The models were not careless — they wrote fluent caveats about interpolation and grid resolution — they never asked whether the request was answerable.

Across the other five tasks, interface changes moved little. Of 18 adjacent contrasts, only the remap result survives Bonferroni correction ($p = 1.5 \times 10^{-4}$, two-sided Fisher exact). Adding evidence on *Area*, 12/20 to 20/20 ($p = 0.0033$), is suggestive but does not survive. Those nulls are by construction: only *Remap* was built to be unanswerable, and the rest ask whether a richer reply helps on a question that already has an honest answer, so flat results there are expected rather than the server failing to add value.

IV. Limitations and Takeaway

This is one server, one library, six small tasks, and 20 runs per cell; only two tasks have an independent exact check, and we do not claim these rates transfer to other domains or servers.

The sharpest open risk is the bypass token. A caller defeats the refusal on purpose by passing `acknowledge` with the string `i-understand-this-result-may-not-be-physical`, which runs the operation and stamps the result unverified. The escape is deliberate — a scientist who knows the extrapolation is fine must be able to proceed — and deliberately awkward. Models rarely reached for it: 16 of 60 runs, 15 from one model, none completing a valid override. But rarely is not never, and whether a determined model can build one and pass extrapolation off as verified is untested.

What generalizes is the mechanism, not the rates. A declared precondition is a scientific artifact rather than an error path, and it performed a check no model performed unprompted. The study measured 5 such checks over 3 operation families; the server now declares 13 over 11, which carry no new runs but show the pattern extending across an API rather than fitting the one case we rigged. For other server authors: state preconditions explicitly, return refusals as structured data rather than prose, and attack your own bypass token before shipping it. The server, the protocol, fixtures, all 480 runs, the scorer with its audit log, and the regeneration scripts are public at `github.com/UXARRAY/uxarray-mcp-server` and `github.com/rajeeja/artifacts_escience_2026_uxarray_mcp`.

Acknowledgment

Supported by the SEATS project, funded by the Regional and Global Modeling and Analysis program area, U.S. Department of Energy Earth and Environmental System Modeling Program, Office of Biological and Environmental Research, DOE Office of Science.

AI assistance was used in drafting this summary and in writing figure code. All content was verified by the authors against the archived runs, and the authors take responsibility for it.