

The Experiment Should Be Allowed to Change the Plan

Lynn Walker | WinMedia

WinMedia Digest essay · Version 1.0 · 5 October 2026

A research program needs room to become smaller. If every result leads to another feature, another formalism, or another language, the experiment has little influence over the architecture. It becomes an occasion to continue a plan already chosen.

Our recent work on computational invention instruments produced two usable software releases. It also narrowed an invention-performance claim and supplied a reason to postpone a new programming language. Those decisions belong in the same account of progress. The work clarified what was supported, what could be released, and where further development had not earned its place.

We organized the effort around four objectives: close a causal-evidence question about Design-Space Calculus, qualify DSC and the Novum language against stable technical contracts, release a bounded public software package, and compare Mandala Core Calculus with a direct implementation in Cog. DSC organizes design states and proposed transformations; Novum works with semantic graphs. Each objective asked for a different kind of evidence. Completing one could not substitute for completing the others.

The first question was whether explicit structure improved AI-generated solutions. Earlier exploratory work had made that possibility attractive. Representing a problem through needs, functions, constraints, resources, mechanisms, candidates, and evidence could make its structure easier to inspect. Explicit transformations could then encourage decomposition, reframing, or reconfiguration. The apparent promise was enough to warrant a controlled test.

The clean trial separated ordinary reasoning, reasoning with an explicit representation scaffold, and representation with added transformation instructions. Twelve frozen problems, three conditions, and four repetitions produced 144 isolated solver runs. Each response received three blinded AI judgments, yielding 432 evaluations. Analysis used the problem as the paired unit; repeated outputs were not counted as additional independent problem domains.

The earlier solution-quality advantage did not replicate. Representation showed a mean change of minus 0.146 on a sixteen-point quality scale relative to ordinary reasoning. Adding transformations showed a further mean change of minus 0.069. Neither contrast established an improvement, and the novelty measures supplied no advantage either. These results constrained the claim that the machinery made the tested model a better inventor.

There was a positive result elsewhere. Representation increased the measured epistemic-integrity score by 0.701 on a four-point scale, with a reported 95 percent interval from 0.521 to 0.875. In this evaluation, that meant stronger rated rigor, failure-mode disclosure, and candor about boundaries. The distinction matters: an answer can become clearer about what remains unsupported while its proposed solution does not become better.

That result deserves attention without being inflated. The trial used one solver model and one evaluator model across twelve problems. It examined prompt scaffolds, not the released DSC or Novum executables. It did not establish general honesty, real-world safety, or superior invention performance. Its protected

inputs and executed judgments remain private, which also limits independent external reproduction. The appropriate response was to narrow the claim and preserve the finding for further study. [1]

Software qualification answered another question. DSC and Novum still needed explicit supported behavior, clean installation, controlled failure paths, and a documented exchange contract. That work remained useful regardless of the invention benchmark. A representation tool can have inspectable behavior and stable interfaces without possessing an experimentally established creativity advantage.

Drawing the release boundary required actual changes. DSC's stable importer reached experimental search modules, so the stable runtime had to be separated before a bounded package could be assembled. Novum likewise needed supported runtime and export behavior separated from research commands. A requested rejection specimen conflicted with its frozen supported subset. The response was to preserve that exclusion rather than quietly enlarge the release claim.

These repairs made the public boundary concrete. The approved result was two developer previews with inspectable supported code, executable examples, citation information, and downloadable artifacts. The public releases are qualified for Python 3.12 only. Their existence establishes a route for outsiders to examine and exercise the software. It does not establish that the pair generates better inventions, and it does not make the private research record publicly reproducible. [2, 3]

The fourth objective asked whether semantic geometry justified a separate programming language. Mandala Core Calculus organizes information through structural stages, concern dimensions, perspectives, and resolution. Those concepts can be useful. The practical question was whether they earned their own language machinery when Cog, our existing execution substrate, could represent the same fixed design problem.

We compared direct Cog with an MCC representation lowered to Cog, using a passive laptop-stand fixture. An early inventory comparison looked favorable to MCC. Closer inspection found differences in supplied object content and resolution levels, including placeholder evaluations. A separate paired representation restored common content before final scoring. Its extra declarations were counted as part of the authoring burden.

With that burden included, MCC used 76 counted structural declaration units against Cog's 86, a reduction of 11.6 percent. Grouping reduced the need to write individual relationships, but both representations expanded to the same fifty dependency endpoints. Adding a Cost dimension required fewer MCC model edits. Inserting a Prototype stage succeeded in direct Cog and was rejected by MCC's fixed stage validator.

The inspection evidence was mixed. MCC named the objective and stages more explicitly, and one independent AI reviewer recovered substantive content from both representations. The native execution traces supplied the same limited process information. Additional MCC defect checks rejected the registered probes, but their counterexamples exposed gaps in the claimed semantic safeguards. Neither readability nor a collection of checker rejections settled the language question by itself.

The final determination was to prefer Cog for this fixture and retain useful geometry as possible schemas or domain helpers. That conclusion applies to a bounded structural comparison: full semantic equivalence remained unproven, the runtime used a research execution bridge, and the candidate narratives were supplied rather than generated. It provides no universal verdict on geometric programming. It does provide a reason to postpone separate Mandala and Yantra language development. [4]

Taken together, the four objectives changed the program's next commitments. We have bounded software that others can inspect. We have an experiment-specific integrity finding worth investigating. We have a

clearer mathematical research baseline, accompanied by a decision to keep its implementation ambitions smaller. The original aspirations remain hypotheses wherever the evidence has not established them.

This work does not prove that our Foundry process outperforms alternative research methods. It offers a concrete account of a process permitting its results to alter the plan. That permission has practical consequences: narrower claims, fewer unsupported development commitments, and usable artifacts with stated limits. Before extending the architecture, we can now ask which unresolved question would actually change what we build next.

Research records

[1] Lynn Walker. [Structured Representation and Epistemic Integrity in AI Reasoning](#). WinMedia technical paper, version 1.0, 5 October 2026. Public account of the private IIE-009/010 report.

[2] [DSC Core v0.2.0rc1](#). Python 3.12 developer preview, released 3 October 2026.

[3] [Novum v0.2.0rc1](#). Python 3.12 developer preview, released 3 October 2026.

[4] Lynn Walker. [Semantic Geometry over Cog](#). WinMedia technical paper, version 1.0, 5 October 2026. Public account of the private MCC-COG-COMP-001 determination of 3 October 2026.

Author note. Prepared with AI-assisted drafting from the cited research records. This essay introduces no new experimental runs or comparative analyses.