Building a Public Research Observatory for High-Scope Open Research

How panta-rhei.site materializes Agenda, Corpus, Results, Verify, Impact, Engage, and artifact release surfaces into one inspectable research system

Thorsten Fuchs • Anna-Sophie Fuchs

May 2026

Version v0.1
Audience Reviewers, contributors, research engineers, open-science practitioners, journalists, and readers inspecting the Panta Rhei public system
Reading time 35–50 minutes
Status Draft technical white paper / blueprint

EXECUTIVE SUMMARY

High-scope open research cannot be made inspectable by publication alone. A preprint, monograph, code repository, static website, or proof library can expose part of a research program. None of those artifacts, by itself, exposes the full public structure needed for mission, obligations, construction, consequences, verification, correction, engagement, and release artifacts.

This white paper presents a technical blueprint for a public research observatory: a static, searchable, citation-ready, versioned, and multi-surface interface for high-scope open research. The Panta Rhei Research Program is used as the implementation case. Its public site organizes the program into eight public lanes: Discover, Program, Agenda, Corpus, Results, Verify, Impact, and Engage, with Publications retained as the stable artifact and release layer.

The paper does not claim that the Panta Rhei scientific framework is validated. It documents how the program has been made inspectable as a public system, and proposes a reusable blueprint for other high-scope open research programs.

Correspondence: press@panta-rhei.site
ORCID: Thorsten Fuchs: 0009-0007-0718-1042; Anna-Sophie Fuchs: 0009-0007-3495-7416

1. INTRODUCTION: BEYOND THE PREPRINT

Open research is often equated with making artifacts available: papers, code, data, notebooks, slides, preprints, or repository snapshots. That is necessary. It is not sufficient for high-scope research.

A high-scope research program must answer many questions at once. What is the program trying to build? What obligations does it accept? What has been constructed? What follows from that construction? What is formally verified? What is empirically testable? What remains unresolved? Where are artifacts released? How can corrections be made? How can reviewers, journalists, contributors, and critics engage responsibly?

A paper cannot do all of this. A repository cannot do all of this. A monograph cannot do all of this. A website can help, but only if it is built as an observatory rather than a landing page.

KEY CLAIM • Core line

The website is not only about the research program. It is the public interface of the research program.

The Panta Rhei website is designed as such an observatory [9]. It exposes orientation, identity, obligations, construction, consequences, inspection, conditional relevance, engagement, and artifacts as distinct public roles.

This white paper is the technical counterpart to the first two press packages. Package 1 defines the inspection standard [8]. Package 2 defines the intellectual category [12]. Package 3 explains how the public system is implemented.

2. DESIGN PRINCIPLE: INTERFACE, NOT BROCHURE

The central design principle is simple:

> The public site is not a brochure for the research. It is the public interface of the research program.

That means the site must do more than introduce a theory. It must expose the program's obligations, constructions, result status, verification routes, release artifacts, corrections, engagement paths, and public relevance boundaries. For high-scope research this is a form of technical and epistemic hygiene. It prevents the program from becoming only a claims page.

The interface must support different reader routes:

- first-contact orientation;
- expert review;
- formal verification;
- result browsing;
- corpus inspection;
- publication and citation;
- public discussion;
- media contact;
- correction reporting.

The same public surface must be useful to a journalist, a mathematician, a software reviewer, a philosopher, an open-science practitioner, a contributor, and a skeptical domain expert. That does not require every page to serve every reader. It requires the system to route readers to the right surface.

3. THE V4 LANE ARCHITECTURE

Panta Rhei's v4 architecture is organized around eight public lanes:

1. Discover;
2. Program;
3. Agenda;

4. Corpus;
5. Results;
6. Verify;
7. Impact;
8. Engage.

Publications remains essential, but it is treated as the artifact and release layer rather than as a primary epistemic lane.

3.1 Discover: orientation

Discover gives first-contact readers a guided route into the system. Its job is not to duplicate the deep content. It routes readers toward Program, Agenda, Corpus, Results, Verify, Impact, Engage, and Publications.

3.2 Program: identity and doctrine

Program defines what the research program is. It owns the canonical statement, coherent-theory-of-reality doctrine, inspection-observatory rationale, founders, scope, status, scrutiny posture, and related-approaches map [10].

3.3 Agenda: obligations

Agenda states what the program binds itself to. It owns Core Semantics, Problem Ledger, Kernel/Model/Reality surfaces, Construction Roadmap, refusals, source policies, and answer-shape discipline [4].

3.4 Corpus: construction body

Corpus is where the theory is built. It exposes definitions, derivations, monographs, registry objects, TauLib projection, construction maps, graph relations, and the Construction Spine as the primary human-readable build route [5].

3.5 Results: consequences

Results is where the built Corpus becomes a world. It presents consequence surfaces: Landmark Results, World Readouts, Problem Ledger Answers, Recovery Target Status, additional derived results, predictions, falsification routes, and progress against the Agenda [11].

3.6 Verify: inspection

Verify is where building becomes accountable. It exposes formal proof checking where formalized, construction-step verification, release manifests, trusted-computing-base disclosure, bridge checks, predictions, falsification paths, assessment protocols, and domain verification [13].

3.7 Impact and Engage

Impact maps what could matter if the work survives inspection [7]. Engage makes openness operational without endorsement: readers can ask questions, report corrections, challenge claims, contribute infrastructure, or route public communication responsibly [6].

4. INFORMATION ARCHITECTURE AS EPISTEMIC ARCHITECTURE

In a normal website, information architecture organizes content. In a public research observatory, information architecture organizes epistemic roles.

Surface	Accountability question
Discover	How does a first-time visitor orient?
Program	What is this research program?
Agenda	What does it bind itself to?
Corpus	What has been built?
Results	What follows from the build?
Verify	How can it be checked?
Impact	Why could it matter conditionally?
Engage	How can others inspect, correct, or participate?
Publications	Where are stable artifacts released?

This turns navigation into inspection architecture. The lane labels are not decorative. They prevent several common collapses.

Agenda must not collapse into Results. Agenda states obligations; Results reports current program stances and consequence readouts.

Corpus must not collapse into Results. Corpus says how the theory is built; Results says what world the built Corpus currently describes.

Verify must not collapse into TauLib. Formal checking can establish internal formal obligations where formalized. It does not, by itself, establish empirical truth, bridge adequacy, semantic correspondence, life recovery, or external acceptance.

Publications must not collapse into the research program. Publications is the artifact shelf: monographs, papers, notes, briefings, white papers, release artifacts, errata, citations, and DOI-ready materials.

5. STATIC, INSPECTABLE, SEARCHABLE

The implementation uses static-site architecture and static search because those choices make the public system easier to inspect.

5.1 Static-site foundation

Jekyll's official documentation describes it as a static site generator that takes text and layouts and creates a static website [2]. For a research observatory, static generation has practical benefits:

- every page can have a canonical URL;
- content can be versioned in Git;
- generated pages can be audited;
- no live database is required to understand public state;
- link structure can be tested before deployment;
- static files can be archived and mirrored;
- search indexing can be deterministic.

Static generation does not make the research correct. It makes the public interface reproducible enough to inspect.

5.2 Static search

Pagefind describes itself as a fully static search library for large sites, with a generated static search bundle and JavaScript search API [3]. For a research observatory, search is not merely a convenience. It is an inspection surface. Readers must be able to jump from terms like "Hubble," "Book V," "TauLib," "falsification," "life," "registry," or "public good" into the relevant public route.

5.3 GitHub substrate

GitHub is not the whole research interface. It is the public substrate for source, formalization, artifact manifests, correction routes, contribution processes, and organization-level orientation. GitHub documentation explains that organization profiles can show public README content and pinned repositories to orient visitors [1]. In this observatory, the organization profile should route readers to the site, TauLib, publications, community channels, and release artifacts.

5.4 DOI and release artifacts

Zenodo documentation explains that DOIs are registered on publication and can be reserved in advance for inclusion in files prior to upload [14]. This package does not reserve a DOI. It uses the same release posture as the previous press packages: metadata is prepared, DOI is forthcoming, and no DOI is claimed until the publication workflow is explicitly approved.

6. DATA AND CONTENT MODEL

A public research observatory should treat content as structured public objects, not only pages.

Program doctrine objects include the canonical statement, coherent theory of reality, inspection observatory, related approaches, scope/status/scrutiny, and red-team posture.

Agenda objects include Core Semantics, Problem Ledger items, source policies, answer-shape requirements, Construction Roadmap steps, refusals, and Kernel/Model/Reality surfaces.

Corpus objects include Construction Spine steps, Registry items, monograph books/parts/chapters, TauLib modules, graph edges, construction maps, and foundational hinges.

Result objects include landmark results, domain world readouts, problem answers, recovery status pages, progress records, predictions, falsification records, and additional derived results.

Verify objects include construction-step verification, Release Manifest, trusted-computing-base disclosure, formal verification surfaces, custom axiom inventory, filter rules, bridge checks, and assessment protocols.

Publication objects include monographs, research papers, research notes, research briefings, white papers, release artifacts, errata, citations, PDF assets, and DOI records.

Engage objects include discussion routes, correction routes, contact routes, support routes, contribution guides, and public review routes.

7. THE REUSABLE OBSERVATORY PATTERN

The pattern can be implemented in ten steps.

1. Define a canonical program statement.
2. Define public lanes by epistemic role, not by artifact type.
3. Separate obligations from results.
4. Separate construction from consequence.
5. Separate formal verification from empirical truth.
6. Make status visible.
7. Provide narrow inspection handles.
8. Keep artifacts stable and citable.
9. Open engagement without endorsement.
10. Make the whole system searchable.

Each step can fail. A site can have a mission but no obligations. It can have obligations but no construction body. It can have construction but no result status. It can have formal proof surfaces but overstate empirical meaning. It can have engagement routes that blur participation and endorsement. The point of the blueprint is to make these failures visible.

8. IMPLEMENTATION CHECKLIST

A high-scope open research observatory should expose:

1. canonical program statement;
2. first-contact orientation lane;
3. identity and doctrine lane;
4. obligation lane;
5. construction lane;

6. result/consequence lane;
7. verification or definition-of-done lane;
8. conditional impact lane;
9. engagement lane;
10. artifact and publication shelf;
11. stable URLs;
12. static or reproducible build process;
13. search;
14. metadata and status labels;
15. source-pinned problem ledgers;
16. construction spine;
17. corpus, registry, and dependency graph;
18. formal or technical verification surface;
19. prediction and falsification routes where applicable;
20. errata and correction mechanism;
21. Media Kit;
22. Review Kit;
23. GitHub or equivalent public substrate;
24. DOI or archive workflow for release artifacts;
25. non-overclaiming language.

This checklist is not a certification standard. It is a practical starting point for making a high-scope research program reviewable.

9. GITHUB ORGANIZATION AS PUBLIC SUBSTRATE

The GitHub organization should not be mistaken for the whole observatory. It is a substrate. It holds source repositories, formalization work, publication manifests, community routes, issue history, and organization-level orientation.

For Panta Rhei, the public organization map can be read as follows.

site. Source for the public observatory. This repository owns the Jekyll pages, layouts, static assets, release-data projections, public routes, and website assertions.

taulib. Lean formalization surface. TauLib can check formalized proof obligations and expose formal status. It does not, by itself, prove empirical truth, bridge adequacy, life recovery, semantic correspondence, or external acceptance.

publications. Artifact and release layer. This repository records PDF assets, publication manifests, hashes, checksums, and catalog files.

research. Exploratory support, scripts, notebooks, reports, and import materials. It can support public reproducibility, but it is not a substitute for a stable result or publication surface.

community. Engagement substrate. It should route questions, corrections, discussion, and participation without implying agreement.

.github. Organization profile and shared community-health files. It orients public visitors to the correct repositories and routes.

This repository map matters because high-scope open research can otherwise scatter itself into disconnected public fragments. The site may explain the research, the proof library may formalize fragments, the publications repository may hold artifacts, and the community repository may invite discussion. The observatory binds those fragments into roles.

10. PUBLICATIONS AS RELEASE SHELF

Publications is not removed from the public system. It is made more precise. In the v4 observatory, Publications is the release shelf. It owns artifacts that need stable citation, licensing, manifest, and download behavior:

- research monographs;
- monograph supplements;
- research papers;
- research notes;
- research briefings;
- white papers;
- numerical ledgers;
- release artifacts;
- errata and citation records.

This separation prevents a common mistake. A publication artifact can be stable and citable without being externally accepted. A PDF can have hashes without being correct. A DOI-ready manifest can support citation without making the content true. A white paper can state a claim boundary without settling the scientific claim behind it.

For that reason, the Package 3 publication page should carry the same discipline as the previous two package pages: abstract, download, citation, DOI status, architecture overview, implementation checklist, journalist route, reviewer route, license, and version history.

11. MEDIA AND REVIEW SURFACES

High-scope research is often misread first through media compression. A public observatory therefore needs media surfaces that are precise enough to prevent accidental overclaiming.

The Package 3 media block should make the safe story clear:

> Panta Rhei is not only publishing claims, papers, or code. It is publishing a public research observatory.

Journalists can responsibly say that Panta Rhei has built a static, searchable, multi-lane public research observatory. They can say the site separates obligations, construction, results, verification, impact, engagement, and publications. They can say GitHub functions as a public substrate for source, formalization, artifacts, and participation routes.

They should not say that the observatory architecture proves the theory, that static publishing validates the claims, that Lean/TauLib verifies empirical results, or that the presence of a DOI or repository implies external acceptance.

The Review Kit has a different job. It should help technical reviewers audit the system before they audit individual claims. A reviewer can ask:

1. Is the canonical program statement visible?
2. Are lanes separated by epistemic role?
3. Is Agenda separated from Results?
4. Is Corpus separated from Results?
5. Is Verify separated from TauLib-only formalization?
6. Are Publications treated as artifacts rather than the whole program?
7. Are URLs stable and canonical?
8. Is the site searchable?
9. Are status labels visible?
10. Are Corpus objects linked to Results and Verify routes?
11. Are formal verification limits stated?
12. Are correction routes visible?

13. Are GitHub repositories role-labeled?
14. Are release artifacts citable or archive-ready?
15. Does the system make narrow inspection possible?

12. CORRECTION AND ENGAGEMENT ROUTES

Open engagement is not the same as agreement. A public observatory should let people participate without endorsing the program.

Engagement surfaces therefore need three properties. First, they must make clear what kind of participation is being invited: question, critique, correction, reproducibility check, domain review, infrastructure contribution, media contact, or institutional dialogue. Second, they must state that participation does not imply endorsement. Third, they must route corrections into a process rather than leaving them as scattered emails or social posts.

For a technical reviewer, correction routes are part of the system's inspectability. If a page overclaims, a link breaks, a source policy is wrong, a PDF hash drifts, a registry item misroutes, or a result status is ambiguous, there must be a public way to report the issue and a maintainable path for recording the fix.

The observatory is therefore not only a publication surface. It is a maintenance surface. It needs changelog, errata, correction, release, and review routes.

13. REVIEWER PATHS

The observatory should support different review paths.

13.1 Architecture review

An architecture reviewer asks whether the public system is inspectable. The route is:

> Homepage -> Program -> Agenda -> Corpus -> Construction Spine -> Results -> Verify -> GitHub -> Publications.

This review does not decide whether the scientific framework is true. It decides whether the public surface makes responsible review possible.

13.2 Formalization review

A formalization reviewer starts in Verify and TauLib. The questions are: which modules exist, which claims are formalized, which axioms are custom, which proof obligations are checked, which dependencies are trusted, and which public claims are not formalized.

13.3 Result review

A result reviewer starts with a concrete result page. The route is:

> Result -> Corpus dependency -> Agenda obligation -> Verify route -> status grammar -> external acceptance boundary.

13.4 Publication review

A publication reviewer starts with a stable artifact. The questions are: does the page link the PDF, does the manifest match the bytes, is the DOI status stated, is the license visible, are source routes linked, and does the artifact avoid overstating review or acceptance?

14. FAILURE MODES

The blueprint is useful only if it also names failure modes.

Claims-page failure. The site presents impressive claims but does not expose obligations, construction, verification, or correction routes.

Repository-fragment failure. Source exists in public repositories, but readers cannot tell which repository owns which role.

Formalization-overread failure. Lean or TauLib checks are read as empirical validation rather than formal verification of scoped objects.

Publication-overread failure. A PDF, manifest, hash, or future DOI is read as external acceptance.

Impact-overread failure. Conditional scenarios are mistaken for achieved deployment, adoption, or public-good delivery.

Engagement-overread failure. Participation is mistaken for agreement or endorsement.

Search-without-structure failure. A site is searchable, but search results do not lead to stable status-marked surfaces.

Naming these failures keeps the blueprint restrained. The observatory is a way to make review possible. It is not a substitute for review.

15. MINIMUM IMPLEMENTATION TOPOLOGY

The minimum topology of a public research observatory is not complicated. The discipline lies in keeping the parts distinct.

15.1 Source layer

The source layer contains repository-tracked material: Markdown pages, data files, layouts, scripts, proof sources, manifests, bibliographies, publication metadata, and generated-source inputs. A reviewer should be able to ask which repository owns a public fact, which file generates a public page, and which commit pinned a release metric.

15.2 Build layer

The build layer turns source into a public artifact. In the Panta Rhei site case this means Jekyll build, Pagefind indexing, link checks, header checks, route assertions, release-metric scanners, and targeted claim-safety assertions. A build failure is not merely a development inconvenience. It is evidence that the public interface has drifted from its declared shape.

15.3 Rendered layer

The rendered layer is what readers inspect: HTML pages, PDF files, search indexes, redirects, static assets, manifests, and downloads. The rendered layer should be crawlable and checkable without privileged access. Important routes should have one H1, stable canonical URLs, correct metadata, and no raw internal data leaks.

15.4 Evidence layer

The evidence layer records whether the rendered public system matches the declared architecture. It includes screenshots, crawl logs, link-check reports, manifest parity checks, publication hashes, audit notes, and review-log closure records. Evidence is not decoration. It is what lets a later reviewer understand why a release was considered fit for publication.

15.5 Engagement layer

The engagement layer gives outsiders a route back into the system: questions, issues, corrections, discussions, review comments, PRs, contact routes, and media/reviewer handoffs. Without this layer, the observatory becomes a static display rather than a reviewable public program.

16. READER QUESTIONS AND OBSERVATORY ANSWERS

A useful observatory should answer reader questions directly.

A first-time reader asks: What is this? The answer should be visible on the homepage and Discover route: an independent open research program dedicated to building a coherent theory of reality.

A doctrine reader asks: What does that category mean? Program should answer through coherent-theory, inspection-observatory, scope/status, and related-approaches pages.

A burden reader asks: What does the program owe? Agenda should expose core semantics, problem ledgers, construction roadmap, source policies, and refusals.

A construction reader asks: What has actually been built? Corpus should expose Construction Spine, monographs, registry objects, TauLib projection, and dependency graph.

A result reader asks: What follows? Results should expose consequence pages with visible status, domain readouts, problem answers, progress records, predictions, and falsification routes.

A verification reader asks: How do I check this? Verify should expose formal proof surfaces, bridge checks, release manifests, construction-step verification, domain verification, and assessment protocols.

A public-relevance reader asks: Why could this matter? Impact should answer conditionally, never as achieved deployment or external acceptance.

A contributor asks: How can I participate? Engage and GitHub should provide routes for questions, critique, correction, review, and infrastructure contribution without implying endorsement.

A citation reader asks: What can I cite? Publications should provide the stable artifact, PDF, license, citation, version, DOI status, and manifest where applicable.

If those questions cannot be answered quickly, the observatory is not yet serving its public function.

17. WHY THE BLUEPRINT IS REUSABLE

The Panta Rhei content is specific, but the observatory pattern is reusable. A different high-scope program could replace Panta Rhei's Corpus, results, formalization stack, domain maps, and publications while keeping the same structural disciplines.

The reusable part is not the lane names by themselves. It is the separation of roles: identity, obligation, construction, consequence, inspection, conditional relevance, engagement, and artifact release. A climate-modeling program, mathematical-foundations program, biomedical knowledge-base project, AI-safety research program, or independent philosophy-of-science project could instantiate those roles differently. The point is not to copy the surface. The point is to avoid publishing high-scope claims without a public inspection path.

The pattern is especially useful when the research object is too large for one paper. A paper can point to the observatory. The observatory can route the reader to the correct paper, code repository, proof module, result page, status legend, correction path, or review checklist.

18. RELEASE DISCIPLINE AND ACCEPTANCE CHECKS

The observatory pattern only works if releases are treated as accountable states, not as continuous editorial drift. A public research site can change quickly, but its public claims should still be inspectable at release boundaries. The practical rule is simple: a release should say what it is, which source commits it represents, which generated projections it includes, which metrics are pinned, which artifacts are current, and which routes a reviewer should inspect.

For Panta Rhei, this discipline is implemented through a combination of source control, generated manifests, page assertions, link checks, release metadata, publication manifests, and post-deployment verification. The important point is not the specific tool chain. The important point is that the public surface is not allowed to become detached from its source structure. If a Corpus export changes, the public page should either update from the export or fail a check. If a release metric changes, the value should move through a pinned manifest rather than through hand-edited prose. If a publication PDF changes, its byte size and hashes should change in the publication manifest. If a route is renamed, the compatibility route should either redirect intentionally or remain as an explicitly marked support surface.

18.1 Minimum release packet

A mature observatory release should be accompanied by a small packet of machine-readable and human-readable evidence:

- the source commits for the public site and relevant source repositories;
- the rendered public site or build output used for QA;
- the release manifest for dynamic public metrics;
- publication manifests for newly released artifacts;
- route assertions for required pages, redirects, headings, and canonical URLs;
- link-check and smoke-test reports;
- focused editorial assertions for claim-safety language;

- visual evidence for the highest-risk public pages; and
- a closure note describing what was verified and what remains outside the release scope.

This packet does not guarantee truth. It guarantees something more modest and more operational: the public state can be reconstructed, challenged, and compared with the declared release boundary.

18.2 Acceptance criteria

The acceptance criteria for an observatory release should separate infrastructure success from epistemic success. A release can pass its observatory checks even while a scientific claim remains contested, partially verified, or externally unaccepted. Conversely, a strong scientific argument can fail as a public observatory if it has no clear source path, no status grammar, no verification route, no correction path, or no stable citation surface.

For a public research observatory, the infrastructure acceptance checks are therefore concrete:

- every primary lane root builds and has one visible title;
- every current public route has a coherent canonical URL;
- generated collections expose source metadata rather than anonymous prose;
- Results pages separate internal program stance from verification state and external acceptance;
- Verify pages separate formal proof, bridge adequacy, empirical checks, and falsification;
- Publications pages distinguish stable artifacts from live Corpus exposition;
- Impact pages remain conditional on upstream survival through Results and Verify; and
- Engage routes make critique, correction, and participation possible without implying endorsement.

Those checks are deliberately plain. They give reviewers something they can test without first accepting the theory. In that sense, release discipline is part of the public argument of the observatory: not an argument that the framework is correct, but an argument that the framework has accepted a public burden of inspection.

18.3 Failure handling

An observatory should also define what happens when checks fail. Broken routes, stale metrics, missing source metadata, ambiguous status labels, and overclaiming language should not be treated as mere website polish. They change the inspection conditions. Some failures are blockers because they prevent a reviewer from finding the relevant source or understanding the claim boundary. Other failures are editorial debt because they make the system less legible without changing the underlying architecture.

The release process should make that distinction explicit. A blocker prevents release or requires immediate remediation. A major issue requires a scheduled fix before public signoff. A polish issue can be logged with evidence and handled in a later wave. This severity grammar prevents two opposite errors: freezing a public research system because every sentence can be improved, or shipping a misleading public architecture because the failure was framed as cosmetic.

In the reusable pattern, failure handling is not an afterthought. It is one of the observatory's core promises. If the architecture invites inspection, it must also give failures a place to land.

19. ARCHITECTURE PLATE NOTE

The visual plate for this package is deferred. Its intended structure is a center spine from Canonical Statement through Program, Agenda, Corpus, Results, and Verify, with side surfaces for Discover, Impact, Engage, Publications, GitHub, Search, Errata, Media Kit, and Review Kit. The caption should say that a high-scope research program becomes inspectable when its obligations, construction, results, verification routes, artifacts, and engagement paths are exposed as one public system.

The absence of this plate in the first Package 3 release is intentional. The white paper and public pages state the architecture now; a future visual plate can make it more legible without changing the claim boundary.

20. WHAT THIS BLUEPRINT DOES NOT CLAIM

This technical blueprint does not prove the Panta Rhei framework. It does not claim that static-site architecture validates scientific results. It does not claim that a GitHub organization, proof assistant, DOI workflow, or search index makes a theory

true.

It claims something narrower: high-scope open research can make itself more inspectable by materializing obligations, construction, consequences, verification routes, artifacts, and engagement paths in a public observatory architecture.

Inspection architecture is not validation. It makes validation, challenge, correction, review, and refusal easier to begin.

21. CONCLUSION

A high-scope open research program should not publish only a claim. It should publish the structure that makes the claim inspectable.

The Panta Rhei Research Program is one implementation attempt. It is published as a static, searchable, multi-lane public research observatory: Discover, Program, Agenda, Corpus, Results, Verify, Impact, Engage, and Publications as the release shelf.

Whether the scientific framework survives review is a separate question. The technical blueprint shows how that review can begin.

HOW TO CITE

Thorsten Fuchs and Anna-Sophie Fuchs, *Building a Public Research Observatory for High-Scope Open Research: How panta-rhei.site materializes Agenda, Corpus, Results, Verify, Impact, Engage, and artifact release surfaces into one inspectable research system*, vo.1, Panta Rhei Research Program, May 2026. DOI forthcoming.

REFERENCES

[1] GitHub Docs. Customizing your organization's profile. https://docs.github.com/en/organizations/collaborating-with-groups-in-organizations/customizing-your-organizations-profile, 2026. Official GitHub documentation; accessed May 2026.

[2] Jekyll. Quickstart. https://jekyllrb.com/docs/, 2026. Official documentation; accessed May 2026.

[3] Pagefind. Pagefind. https://pagefind.app/, 2026. Official project documentation; accessed May 2026.

[4] Panta Rhei Research Program. Agenda. https://panta-rhei.site/program/research-agenda/, 2026. Canonical Agenda lane.

[5] Panta Rhei Research Program. Corpus. https://panta-rhei.site/corpus/, 2026. Canonical Corpus lane.

[6] Panta Rhei Research Program. Engage. https://panta-rhei.site/engage/, 2026. Canonical Engage lane.

[7] Panta Rhei Research Program. Impact. https://panta-rhei.site/impact/, 2026. Canonical Impact lane.

[8] Panta Rhei Research Program. Inspection architecture for high-scope open research. https://panta-rhei.site/publications/white-papers/inspection-architecture-high-scope-open-research/, 2026. Press Package 1 white paper.

[9] Panta Rhei Research Program. Panta rhei research program. https://panta-rhei.site/, 2026. Canonical public website.

[10] Panta Rhei Research Program. Program. https://panta-rhei.site/program/, 2026. Canonical Program lane.

[11] Panta Rhei Research Program. Results. https://panta-rhei.site/results/, 2026. Canonical Results lane.

[12] Panta Rhei Research Program. The shape of a theory of reality. https://panta-rhei.site/publications/white-papers/the-shape-of-a-theory-of-reality/, 2026. Press Package 2 white paper.

[13] Panta Rhei Research Program. Verify. https://panta-rhei.site/verify/, 2026. Canonical Verify lane.

[14] Zenodo. Digital object identifier (doi). https://help.zenodo.org/docs/deposit/describe-records/reserve-doi/, 2026. Official Zenodo documentation; accessed May 2026.