Info: What Every Proposal Must Contain

Document Number:	P4133R0
Date:	2026-03-21
Intent:	Inform
Audience:	WG21

Reply-to:	Vinnie Falco vinnie.falco@gmail.com C++ Alliance Proposal Team

Table of Contents

Abstract

The committee measures whether the train runs on time. It does not measure whether the passengers arrived at the right destination.

WG21 tracks process metrics - meetings held, polls taken, standards shipped on schedule. It does not track outcome metrics - whether adopted features achieved their claimed benefits, whether predictions made during adoption materialized, or whether the cost of standardization was justified over ecosystem delivery. This paper proposes what a healthy feedback loop would contain for both library and language proposals, maps each element to the failure modes documented in P4050R0[1], and audits the published record to measure how close the committee comes. The evidence is public. The gap is the reader's to measure.

Revision History

R0: March 2026 (post-Croydon mailing)

- Initial version.

---

1. Disclosure

The author has papers in the WG21 mailing and participates in the committee process. The author developed and maintains Corosio[2] and Capy[3] and is a co-author of P2469R0[4]. The author's preferred asynchronous model competes with P2300R10[5] (`std::execution`).

This paper is a companion to P4050R0[1], which identifies fourteen failure modes and proposes proportional deliberation. That paper asks how the process should scale with consequence. This paper asks a different question: does the process evaluate its own output at all?

This paper is also a companion to P4034R0[6] (WG21-SAGE), which proposes capturing expert judgment through structured interviews. That paper addresses input quality - the knowledge that goes into decisions. This paper addresses output measurement - whether the decisions produced the claimed results.

The analysis is structural, not personal. Every decision examined was made by experienced practitioners under real constraints. The intent is to document what the published record contains and what it does not.

---

2. The Two Kinds of Measurement

Engineering organizations measure two things: whether the process was followed, and whether the output was good.

Process metrics answer: Did we hold the meetings? Did we take the polls? Did the standard ship on schedule? Did papers receive review? These are necessary. They are not sufficient.

Outcome metrics answer: Did the adopted feature achieve its claimed benefits? Did users adopt it? Did implementers ship it on time? Did the predictions made during adoption hold? Were the costs justified? These close the loop. Without them, the process runs open-loop - it produces output but never checks whether the output was correct.

P1000R6[7] established the three-year release cadence. The cadence is a process metric. It measures whether the train departs on schedule. It does not measure whether the cargo was worth shipping.

P2000R5[8] articulates high-level direction: "standardize existing practice," "language support for library." These are input guidelines. They do not include a mechanism for checking whether the committee followed them, or whether following them produced good outcomes.

P4034R0[6] states: "The committee has no retrospectives, no formal onboarding, no written institutional memory."

The process metrics exist. The outcome metrics do not.

---

3. What a Library Proposal Should Contain

Standardization has costs that ecosystem delivery does not bear. An ecosystem library can fix bugs in the next release, change its API when users report friction, drop a feature that turned out to be a mistake, and evolve its documentation as understanding deepens. A standard library feature is frozen at the point of adoption. It must be implemented by every conforming vendor. It must be taught to every student. It must be maintained in perpetuity. It must be ABI-stable across decades.

These costs are real. A proposal that does not justify them against the alternative of ecosystem delivery has not made its case. "We find this useful" is not justification. "This cannot be delivered by the ecosystem because X, and standardization provides Y that ecosystem delivery cannot" is justification.

The following artifacts constitute the minimum evidence that a library proposal is ready for committee time.

3.1 Evidence of Need

The proposal must demonstrate why standardization provides more value than ecosystem delivery. The demonstration must be specific:

- What problem does this solve that the ecosystem cannot?
- What property of standardization (portability, guaranteed availability, ABI stability, teaching curriculum inclusion) is required?
- What is the cost of standardization (implementer burden, ABI commitment, teaching load, maintenance in perpetuity) and why does the benefit exceed it?

A proposal that cannot answer these questions belongs in the ecosystem, where it can evolve freely and prove itself before bearing the costs of standardization.

3.2 Complete Implementation

A complete library with benchmarks, unit tests, and documentation. Not a sketch. Not a header with stubs. Not a paper that describes an API without building it. A library that compiles, passes its tests, and demonstrates the design.

The committee historically accepted proposals without implementations because producing them was expensive. That excuse no longer holds. Section 5 documents why.

3.3 Steel Man Against Standardization

The proposal must contain the strongest argument for why it should NOT be standardized - why the ecosystem might be enough. The proposal must then confront that argument and defeat it with evidence.

A proposal that does not contain this section has not considered the alternative. The committee member who raises the objection in the room is doing work the author should have done in the paper. If the author cannot defeat the argument against standardization, the proposal is not ready.

3.4 Steel Man of Competing Designs

The proposal must contain the strongest case for alternative designs that solve the same problem differently. What do they provide that this design does not? What are their advantages? Why was this design chosen over them?

P4094R0[9] documents that three deployed executor models were unified into P0443R0[10] with no analysis of what each domain lost. P4098R0[11] documents six unification claims supported by one hypothetical code snippet. A proposal that does not steel man the competition has not demonstrated that it examined the design space.

3.5 Post-Adoption Metrics

What does success look like? The proposal must define, before adoption, how the committee will know whether the feature worked. Possible metrics:

- Deployment rate across major standard library implementations
- User adoption surveys
- Defect rates in the first two releases after adoption
- Teaching difficulty reported by educators
- Frequency of workarounds in user code

A feature adopted without success criteria cannot be evaluated after adoption. The feedback loop is open before it begins.

3.6 Forced Retrospective Trigger

A mandatory look-back after a defined interval - two releases, or six years, whichever comes first. The retrospective asks:

- Did the claimed benefits materialize?
- Did the predictions made during adoption hold?
- Did users adopt the feature?
- Did implementers ship it on time?
- What was learned that the next proposal should know?

P4034R0[6] documents that WG21 has no retrospectives. A forced trigger ensures that the loop closes whether or not anyone volunteers to close it.

3.7 Decision Record

Why this design, not that one. What was traded away. What alternatives were considered and why they were rejected. Under what conditions the decision should be revisited.

P4050R0[1] Section 2 documents the current state: "The reasoning behind the result is not recorded. The alternatives considered are not recorded. The dissenting views are not recorded. The conditions for revisiting the decision are not recorded."

3.8 Domain Coverage Attestation

Which domains were represented in the room when the poll was taken. P4097R0[12] documents the October 2021 poll where "including networking" achieved consensus while published voter comments included "can't judge its suitability for networking" from voters who voted Weakly Favor. Domain coverage is metadata on the record. It does not weight votes. It tells a future reader whether the affected domains were heard.

3.9 Prediction Registry

Claims made during adoption - "this will enable X," "this covers networking," "this will be implementable in N months" - recorded with falsifiable criteria and a revisit date. P4098R0[11] surveys published claims that shaped a decade of committee decisions. For most claims, the published record contains no follow-up. A prediction registry ensures that claims are tested, not just made.

---

4. What a Language Proposal Should Contain

Language features cannot be delivered by the ecosystem. The justification bar shifts: the question is not "why standardize instead of leaving it in the ecosystem?" but "does the benefit justify the permanent complexity added to the language?" The costs are different - every compiler must implement it, every teaching curriculum must cover it, every user must understand it when reading code - but the need for evidence is the same.

4.1 Evidence of Need

The proposal must demonstrate that the problem cannot be solved with existing language facilities, or that the existing solution is so costly in practice that language support is justified. The demonstration must include real-world code showing the current pain and the proposed relief.

4.2 Proof-of-Concept Implementation

A proof-of-concept implementation in a compiler fork or branch. The implementation does not need to be production quality. It needs to demonstrate that the feature works, that the edge cases have been considered, and that the interaction with existing features has been explored.

A language proposal without an implementation is a wish, not a proposal. Section 5 documents why this is now feasible for any proposal whose complexity is appropriate for standardization.

4.3 Teaching Story

How does this feature interact with the existing language for a student encountering it for the first time? What does the error message look like when the feature is misused? What existing mental models does it reinforce, and which does it break?

4.4 Steel Man Against Adoption

The strongest case for why this feature should NOT be added to the language. What complexity does it add? What teaching burden does it impose? What interaction with existing features creates surprising behavior? The proposal must confront these arguments and defeat them with evidence.

4.5 Steel Man of Competing Designs

The strongest case for alternative designs - including the design of doing nothing. If the problem can be solved with a library, why does it need language support? If another language feature could solve it differently, why is this design better?

4.6 Remaining Checklist Items

The remaining items from Section 3 apply without modification: post-adoption metrics (Section 3.5), forced retrospective trigger (Section 3.6), decision record (Section 3.7), domain coverage attestation (Section 3.8), and prediction registry (Section 3.9).

---

5. The AI Capability Shift

Before 2023, producing a complete library implementation was months of skilled labor. A proof-of-concept compiler fork was a research project. The evidence bar was calibrated to that cost. The committee accepted proposals without implementations because producing them was expensive relative to the value of the evidence they provided.

Frontier generative AI models have collapsed the cost of producing proof-of-concept code by an order of magnitude or more. The quality bar for this evidence is deliberately low. It does not need to be production-ready, mission-critical code. It needs to compile, pass tests, and demonstrate the design.

5.1 Library Proposals

A complete library with benchmarks, unit tests, and documentation is now achievable in days with AI assistance, not months. The library does not need to be optimized for production deployment. It needs to demonstrate that the API works, that the edge cases compile, that the performance characteristics are in the expected range, and that the documentation is coherent. A proposal that arrives without this has not done the bare minimum.

5.2 Language Proposals

A proof-of-concept compiler fork is now feasible for any proposal whose complexity is appropriate for standardization. Generative AI can assist with the mechanical aspects of compiler modification - AST node definitions, parser extensions, template instantiation paths, diagnostic messages. The author provides the design. The AI provides the scaffolding.

If a language proposal is so complex that the author cannot produce even a proof-of-concept implementation with AI assistance, that complexity is itself evidence. It suggests that the feature may impose implementation costs that the committee has not evaluated, or that the design has not been thought through to the level of detail that implementation requires.

5.3 The Excuse of Cost Is Gone

The committee historically calibrated its evidence bar to the cost of producing evidence. That calibration is now stale. The cost has dropped. The bar should rise to match.

This is not an argument about AI in the committee process - P4023R0[13] addresses that question. This is a simpler observation: the cost of producing evidence has dropped. A proposal without an implementation is no longer a proposal that could not afford one. It is a proposal that chose not to produce one.

---

6. What the Process Should Contain

The proposal-level checklists in Sections 3 and 4 address individual papers. The process-level requirements address the committee's own workflow.

6.1 Retrospective Mechanism

A scheduled review of adopted features against their original claims. The review is triggered automatically - by time (two releases after adoption) or by external signal (implementer reports, user surveys, defect data). The review asks the questions in Section 3.6 and publishes the answers as a committee document.

6.2 Symmetric Evidence Bar

The same standard of evidence applied to the incumbent and the challenger. P4050R0[1] failure mode A4 documents that P2464R0[14] stated "I don't know" whether Networking TS compositions scale. Five years later, no sender-based networking has shipped either. The constraint applied to one model was not applied symmetrically to the replacement. Failure mode E2 documents the same pattern at the claims level: the GPU and infrastructure evidence is real, the networking evidence column is empty, and both categories shaped committee decisions.

6.3 Knowledge Continuity

Design rationale that survives author turnover. P4050R0[1] failure modes B1 and B2 document the mechanism of loss: the continuation framing was established in P0113R0[15] (2015), carried by institutional knowledge rather than by the API surface, and dropped out when the property hint was removed in 2019 by authors who inherited the API but not the conceptual model. The decision record (Section 3.7) is the artifact that prevents this.

6.4 Outcome Tracking

Did the standard ship? Did implementers ship? Did users adopt? P4131R0[16] documents the train model's claims against outcomes for each release from C++14 through C++26. The train model promised that features would be pulled when not ready. The record shows they are pushed when almost ready. Outcome tracking makes this visible before the pattern compounds.

---

7. Mapping to Documented Failure Modes

The ideal model in Sections 3 through 6 is not invented from scratch. It is the inverse of P4050R0[1]'s seventeen checklist items. Each element of the ideal model exists because its absence is a documented failure mode. The following table maps each failure mode to the corresponding requirement.

P4050 #	Failure Mode	Ideal-Model Requirement
A1	Architecture without working demonstration	Complete implementation (3.2, 4.2)
A2	Assertions without deployed evidence	Evidence of need + standardization cost justification (3.1, 4.1)
A3	Bundled-domain consensus	Domain coverage attestation (3.8)
A4	Predictions without follow-up	Forced retrospective + prediction registry (3.6, 3.9)
B1	Rationale lost across paper boundaries	Decision record + knowledge continuity (3.7, 6.3)
B2	Terminology drift erasing a conceptual model	Decision record (3.7)
C1	Coupling that blocks independent delivery	Scope independence (implicit in evidence of need)
C2	Iteration without deployment	Complete implementation - AI capability shift eliminates cost excuse (5)
D1	One-framing analysis	Steel man competing designs (3.4, 4.5)
D2	Pivot without domain-cost evaluation	Steel man against standardization + domain coverage (3.3, 3.8)
D3	Affected stakeholders absent	Domain coverage attestation (3.8)
D4	API limits mistaken for domain limits	Steel man competing designs (3.4, 4.5)
E1	Proceeding without consensus	Decision record + dissent record (3.7)
E2	Non-uniform evidence bar	Symmetric evidence bar (6.2)
F1	Missing artifacts disproportionate to decision	The full checklist itself (Sections 3-4)
G1	Decision without domain coverage	Domain coverage attestation (3.8)
G2	Absent stakeholder on consequential decision	Domain coverage attestation + stakeholder presence (3.8)

The ideal model is the minimum set of artifacts whose absence has been demonstrated to produce failure.

8. The Audit

This section examines the published record for each element of the ideal model. The evidence is drawn from published papers, standing documents, meeting minutes, and public testimony. Where the published record contains the element, it is documented. Where the published record does not contain it, the cell is empty.

8.1 Evidence of Need / Standardization Justification

P2000R5[8] states: "Prefer to standardize existing practice." SD-9[17] documents default positions on `[[nodiscard]]`, `noexcept`, and similar policy questions. Neither document requires a cost/benefit analysis of standardization versus ecosystem delivery. Neither asks: "Why does this need to be in the standard rather than in a library on GitHub?"

SD-7[18] describes how to format and submit a paper. It does not describe what evidence a paper must contain.

The published record contains guidance on what to standardize ("existing practice") but no requirement to justify the cost of standardization over the alternative of ecosystem delivery.

8.2 Complete Implementation

The published record contains examples on both sides.

With implementation: C++20 coroutines had MSVC shipping TS support in 2016 and Clang in 2017, years before the C++20 merge[19]. Ranges had range-v3 deployed before standardization. These features had implementation experience.

Without implementation: P4094R0[9] documents that P0443R0[10] unified three executor models with no prototype demonstrating the unified model working across all three domains. The unified model was never deployed as unified. It was replaced by P2300R10[5]. P4098R0[11] documents six rationale assertions for unification supported by one hypothetical code snippet.

The published record does not require a complete implementation as a prerequisite for committee time. Some proposals arrive with implementations. Some do not. The committee proceeds in both cases.

8.3 Steel Man Against Standardization

The published record was searched for any proposal that contains a structured section arguing why the feature should NOT be standardized - why the ecosystem might be enough - and then defeating that argument with evidence.

No such requirement exists in SD-7, SD-9, P2000R5, or any standing document. Individual papers occasionally address the question informally. It is not a required artifact.

8.4 Steel Man of Competing Designs

P2000R5[8] warns against opposing a proposal "because it is seen as competing with your favorite proposal for time/resources." This addresses voter behavior. It does not require the proposal author to document competing designs.

P4094R0[9] documents that the unification of executors proceeded without evaluating what each domain lost. P4098R0[11] Section 2.1 documents six unification claims with no competing-design analysis. P4050R0[1] failure mode D1 documents that P1525R0[20]'s pivotal analysis examined only one framing.

The published record does not require proposals to steel man competing designs.

8.5 Post-Adoption Metrics

The published record was searched for any document that defines success criteria for an adopted feature - deployment targets, user adoption thresholds, defect rate expectations, teaching difficulty benchmarks.

P1000R6[7] defines the release cadence. It does not define what "ready" means beyond "has completed LWG/CWG review." P2000R5[8] articulates direction. It does not define how to measure whether the direction was followed or whether following it produced good outcomes.

No post-adoption metrics exist in the published record.

8.6 Forced Retrospective

P4034R0[6]: "The committee has no retrospectives, no formal onboarding, no written institutional memory."

The published record was searched for any mandatory look-back mechanism - a standing document, a process requirement, a scheduled review of adopted features against their original claims.

No forced retrospective mechanism exists. Retrospectives occur only when an individual author volunteers to write one. The five retrospectives in P4094R0[9] through P4098R0[11] were written by the author of this paper, not by the committee. They are the exception that illustrates the absence of the rule.

8.7 Decision Records

P4050R0[1] Section 2:

> *The chair calls the poll. The result is recorded: SF:24 / WF:16 / N:3 / WA:6 / SA:3. The reasoning behind the result is not recorded. The alternatives considered are not recorded. The dissenting views are not recorded. The conditions for revisiting the decision are not recorded.*

The published record contains poll numbers. It does not contain decision rationale, alternatives considered, dissenting views, or revisit conditions.

8.8 Domain Coverage

P4097R0[12] documents the October 2021 poll "The sender/receiver model (P2300) is a good basis for most asynchronous use cases, including networking" (SF:24 / WF:16 / N:3 / WA:6 / SA:3). Published voter comments included "can't judge its suitability for networking" from voters who voted Weakly Favor[21].

The published record does not attach domain coverage information to poll results. A poll of one hundred generalists and a poll of five domain practitioners produce the same output format.

8.9 Prediction Registry

P4098R0[11] surveys published claims that shaped the trajectory of executors, networking, and asynchronous programming. For each claim, the published record was searched for supporting evidence. The results:

Category	Claims surveyed	Claims with deployed evidence	Claims with hypothetical evidence only	Claims with no evidence found
Unification (2014-2020)	6	1	1	4
Basis operation (2019)	4	0	3	1
Networking dependency	4	2	0	2
P2464R0 diagnosis	4	1	0	3
Networking claim (2021)	3	0	1	2
P2300 deployment	4	3	0	1
Pro-Networking-TS	4	3	0	1

No mechanism exists in the published record to register predictions at the time of adoption and revisit them at defined intervals. P4050R0[1] failure mode A4 documents the consequence: predictions are made, acted upon, and never followed up.

8.10 Symmetric Evidence Bar

P4050R0[1] failure mode A4: P2464R0[14] stated "I don't know" whether Networking TS compositions scale. Five years later, no sender-based networking has shipped either. The constraint applied to one model was not applied symmetrically to the replacement.

P4050R0[1] failure mode E2: The GPU and infrastructure evidence for sender/receiver is real. The networking evidence column is empty for most claims. Both categories shaped committee decisions with the same weight.

No mechanism exists in the published record to ensure that the same evidence bar is applied to each side of a directional decision.

8.11 Knowledge Continuity

P4050R0[1] failure mode B1: The continuation framing was established in P0113R0[15] (2015). It was carried by institutional knowledge rather than by the API surface or the type system. When the property hint was removed in 2019 by authors who inherited the API surface but not the conceptual model, the framing dropped out.

P4050R0[1] failure mode B2: `dispatch`/`post`/`defer` became `execute(F&&)` across seven papers over five years. Each rename was locally reasonable. The cumulative effect was that the continuation framing was no longer visible on the API surface.

No mechanism exists in the published record to ensure that design rationale survives author turnover.

8.12 Outcome Tracking

P4131R0[16] documents the train model's claims against outcomes. The model promised "ship what's ready" and a safety valve: "If we see a feature is not ready yet, we pull it back out." The record shows features pushed when almost ready rather than pulled when not ready.

P4129R1[22] documents attendance-dependent outcome reversals, poll wording effects, and the disproportionate weight of SA votes under the consensus threshold. These are structural properties of the voting mechanism. They are observable in the published record. No committee document analyzes them.

The committee tracks whether the standard shipped on time. It does not track whether the features in the standard achieved their claimed benefits.

---

9. The Scorecard

#	Ideal-Model Element	P4050 Failure Mode	Present in Published Record?	Source
1	Evidence of need	A2	Guidance ("existing practice") but no cost/benefit requirement	P2000R5, SD-9
2	Complete implementation	A1, C2	Sometimes (coroutines, ranges). Not required.	P4094R0, P4098R0
3	Steel man against standardization	D2	Not found as a required artifact	SD-7, SD-9
4	Steel man competing designs	D1, D4	Not found as a required artifact	P4094R0, P4098R0
5	Post-adoption metrics	A4	Not found	P1000R6, P2000R5
6	Forced retrospective	A4	Not found. "No retrospectives."	P4034R0
7	Decision record	B1, B2, E1	Poll numbers only. No rationale, alternatives, dissent, or revisit criteria.	P4050R0 Section 2
8	Domain coverage	A3, D3, G1, G2	Not attached to poll results	P4097R0
9	Prediction registry	A4	Not found. Claims made and not followed up.	P4098R0
10	Symmetric evidence bar	A4, E2	Not found. Evidence bar applied non-uniformly.	P4050R0
11	Knowledge continuity	B1, B2	Not found. Rationale lost across paper boundaries.	P4050R0
12	Outcome tracking	(process-level)	Process metrics exist. Outcome metrics do not.	P1000R6, P4131R0

Of twelve ideal-model elements, one is partially present (evidence of need, as informal guidance), one is sometimes present (complete implementation, for some proposals), and ten are absent from the published record.

---

10. Directions

The following directions are presented as options, not recommendations. The committee evaluates the tradeoffs. Each option addresses one or more gaps in the scorecard.

Option A: Require Implementation for Committee Time

Library proposals without a complete implementation (benchmarks, unit tests, documentation) are deferred until the implementation exists. Language proposals without a proof-of-concept compiler fork are deferred. The chair applies the requirement using existing scheduling power. No new standing document is needed.

Addresses: Scorecard items 2 (complete implementation).

Option B: Require a Standardization Justification Section

SD-7 is amended to require a section in every proposal answering: "Why does this need to be in the standard rather than in the ecosystem?" The section must include the steel man against standardization and the steel man of competing designs.

Addresses: Scorecard items 1 (evidence of need), 3 (steel man against standardization), 4 (steel man competing designs).

Option C: Adopt the P4050 Checklist

The seventeen-item checklist from P4050R0[1] Section 9 is adopted as a standing document. The chair uses it to assess readiness before scheduling consequential decisions. The checklist scales with the decision tier - routine decisions require only the paper, structural decisions require the full artifact set.

Addresses: Scorecard items 7 (decision record), 8 (domain coverage), 10 (symmetric evidence bar).

Option D: Institute Forced Retrospectives

A new standing document requires a retrospective paper for every feature adopted into the standard, published no later than two releases after adoption. The retrospective is assigned to the original paper author or, if unavailable, to a volunteer. The retrospective asks the questions in Section 3.6.

Addresses: Scorecard items 5 (post-adoption metrics), 6 (forced retrospective), 9 (prediction registry), 12 (outcome tracking).

Option E: Attach Domain Coverage to Poll Records

Poll results include a domain coverage annotation: "Of N self-registered domain practitioners, the distribution was X." The annotation is metadata, not a weighting function. The committee interprets it. A future reader can see whether the affected domains were heard.

Addresses: Scorecard item 8 (domain coverage).

Option F: Publish Decision Rationale for Tier 3+ Decisions

For decisions sized at Tier 3 or above under P4050R0[1]'s framework, the chair or a designated recorder publishes a decision rationale document: reasoning, alternatives considered, dissenting views, and conditions for revisiting the decision.

Addresses: Scorecard items 7 (decision record), 11 (knowledge continuity).

---

11. Anticipated Objections

Q: This adds bureaucracy to an already slow process.

A: The bureaucracy scales with the consequence. Routine proposals (Tier 1 under P4050R0[1]) require only the paper and an implementation. The additional artifacts are required only for consequential decisions - the decisions where the cost of getting it wrong is measured in years, not meetings. The cost of not producing these artifacts is documented in P4094R0[9] through P4098R0[11]: twenty-one years without networking in the C++ standard.

Q: Requiring implementations will slow down proposals.

A: It will slow down proposals that do not have implementations. That is the point. A proposal without an implementation has not demonstrated that its design works. The committee's time is finite. Proposals that have done the work should receive that time before proposals that have not.

Q: Not every author has the resources to produce a complete implementation.

A: Section 5 addresses this. Generative AI has collapsed the cost of producing proof-of-concept code. A complete library with benchmarks, unit tests, and documentation is now achievable in days with AI assistance. A proof-of-concept compiler fork is feasible for any proposal whose complexity is appropriate for standardization. The excuse of cost is gone.

Q: Some important features were standardized without implementations and turned out fine.

A: Some did. Some did not. P0443[10] went through fourteen revisions without deployment and was replaced. The question is not whether it is possible to succeed without an implementation. The question is whether requiring one would have caught the failures earlier.

Q: This paper is arguing for the author's own library.

A: Section 1 discloses this. The ideal model applies to every proposal, including the author's. The failure modes are extracted from the published record. The scorecard grades the process, not any individual proposal. The ideal model and the audit stand on the sources, not on who assembled them.

Q: The committee already evaluates proposals carefully.

A: The committee evaluates proposals. It does not evaluate outcomes. The scorecard in Section 9 documents what the published record contains and what it does not. Ten of twelve ideal-model elements are absent. The committee's evaluation of proposals may be thorough. Its evaluation of whether those proposals achieved their goals after adoption does not exist.

Acknowledgements

The author thanks Joaquín M López Muñoz and Peter Dimov for their critique of P4129R1[22], which informed the analytical approach used here. The author thanks Steve Gerbino for feedback on the ideal model.

References

- [1] P4050R0: "Failure Modes in Large-Scale Standardization." Falco. https://wg21.link/p4050r0
- [2] Corosio: https://github.com/cppalliance/corosio
- [3] Capy: https://github.com/cppalliance/capy
- [4] P2469R0: "Response to P2464: The Networking TS is baked, P2300 Sender/Receiver is not." Kohlhoff, Allsop, Falco, Hodges, Morgenstern. https://wg21.link/p2469r0
- [5] P2300R10: "`std::execution`." Dominiak, Evtushenko, Baker, Teodorescu, Howes, Shoop, Garland, Niebler, Lelbach. https://wg21.link/p2300r10
- [6] P4034R0: "Before They Retire: WG21-SAGE." Falco. https://wg21.link/p4034r0
- [7] P1000R6: "C++ IS schedule." Sutter. https://wg21.link/p1000r6
- [8] P2000R5: "Direction for ISO C++." Stroustrup, Hinnant, Orr, Vandevoorde, Wong. https://wg21.link/p2000r5
- [9] P4094R0: "Retrospective: The Unification of Executors and P0443." Falco. https://wg21.link/p4094r0
- [10] P0443R0: "A Unified Executors Proposal for C++." Hoberock, Garland, Kohlhoff, Mysen, Edwards. https://wg21.link/p0443r0
- [11] P4098R0: "Retrospective: Async Claims and Evidence." Falco. https://wg21.link/p4098r0
- [12] P4097R0: "Retrospective: The Networking Claim and P2453R0." Falco. https://wg21.link/p4097r0
- [13] P4023R0: "Strategic Direction for AI in C++." Directions Group. https://isocpp.org/files/papers/p4023r0.pdf
- [14] P2464R0: "Ruminations on networking and executors." Voutilainen. https://wg21.link/p2464r0
- [15] P0113R0: "Executors and Asynchronous Operations, Revision 2." Kohlhoff. https://wg21.link/p0113r0
- [16] P4131R0: "Retrospective: Effects of the WG21 Train Model." Falco. https://wg21.link/p4131r0
- [17] SD-9: "Library Evolution Policies." https://isocpp.org/std/standing-documents/sd-9-library-evolution-policies
- [18] SD-7: "Mailing Procedures and How to Write Papers." https://isocpp.org/std/standing-documents/sd-7-mailing-procedures-and-how-to-write-papers
- [19] P0912R5: "Merge Coroutines TS into C++20." Nishanov. https://wg21.link/p0912r5
- [20] P1525R0: "One-Way execute is a Poor Basis Operation." Niebler, Shoop, Baker, Howes. https://wg21.link/p1525r0
- [21] P2453R0: "2021 October Library Evolution Poll Outcomes." Lelbach, Fracassi, Craig. https://wg21.link/p2453r0
- [22] P4129R1: "The Dynamics of Voting in WG21." Falco. https://wg21.link/p4129r1