Implementation reality of WG21 standardization

Document number: P3962
Date: 2026-01-13
Audience: All of WG21
Authors :
Nina Ranns
Erich Keane
Vlad Serebrennikov
Aaron Ballman
Iain Sandoe
Jonathan Caves
Cameron DaCamara
Gabriel Dos Reis
Gonzalo Brito
Christof Meerwald
Chuanqi Xu
Shafik Yaghmour
Cody Miller
Wyatt Childers
Waffl3x (Alex)
Bruno Cardoso Lopes
Hubert Tong
Louis Dionne

1. Introduction

At the Kona 2025 WG21 meeting, a group of implementers got together to discuss their challenges. We had 20 in person attendees and 9 remote attendees. The group spent an evening airing out their thoughts. We deliberately kept the meeting closed and encouraged being open, honest, and unfiltered in hope of creating a safe space for everyone to feel heard. This document is a summary of that discussion, documenting the general types of concerns raised, and then following with suggestions on how to tackle them.

2. Topics discussed

This is not an extensive list of topics discussed. For brevity, only the major topics have been included.

2.1 Cost, resources, and economic realities

The group highlighted the need for greater visibility and shared understanding of the real costs associated with features. While individual proposals are often evaluated on technical merit and basic feasibility, the cumulative cost of implementing, maintaining, testing, and supporting features over time is less visible in committee discussions.

Compiler and library development operates under significant resource constraints. Many implementers are volunteers or are only partially funded for standardisation work, and there is often no dedicated staffing to implement all standardized features. As a result, full conformance to recent standards remains difficult in practice, with some implementations still working toward C++20 conformance with limited capacity to adopt newer standards. Not having a certain feature available from all vendors makes the standard non-portable and lowers the chance of adoption.

Participants emphasized that implementation cost is not limited to initial development. Ongoing maintenance, performance implications, ABI stability, testing matrix growth, and interactions with existing features all contribute substantially to the long-term burden.

There was broad agreement that clearer acknowledgment of cost and trade-offs would improve decision-making. Additionally, it should be understood that adding new features to the standard necessarily displaces other work, including bug fixes, conformance improvements, performance tuning, and various high-value library enhancements.

2.2 Implementer voice and understanding implementation complexity

The groups expressed concern that implementers' role and expertise is not always fully reflected in committee processes, particularly during early design discussions. Implementation feedback is often introduced late, treated as adversarial, or framed primarily as an obstacle to progress rather than as essential design input. Strengthening the visibility and influence of implementer perspectives earlier in the process may help ensure that designs converge on solutions that are both technically sound and practically deployable.

Several participants noted that there is sometimes an implicit assumption that proposals are “straightforward” to implement, based on mental models that do not reflect the realities of large, mature codebases. In practice, seemingly small or localized changes can require extensive refactoring, interact in unexpected ways with existing features, or impose significant testing and maintenance costs. Distinguishing between theoretical possibility and practical plausibility was identified as an important lens that is not consistently applied today.

There was also concern that committee discussions impact upon areas of the toolchain (especially the intermediate representation optimisers) where relevant expertise is not well-represented by regular committee attendees. This does on occasion result in implementation issues that were not addressed in the proposal.

Participants noted that what is often presented as “implementation experience” does not always provide sufficient information to assess real-world feasibility. Partial prototypes, local forks, or proof-of-concept implementations can be useful for exploring ideas, but they rarely reflect the effort required to upstream, maintain, test, and deploy a feature across large and diverse codebases. Similarly, a prototype based on a particular implementation can’t always be extrapolated to all implementations as the cost of a feature may be widely different on different implementations. Additionally, while we encourage implementation experience for new features, the room also observed that we should do more as a committee to make implementation experience a requirement for new features.

Overall, participants emphasized that implementers are not merely responsible for wording or post-hoc execution, but for turning the standard into something coherent, performant, and usable in real systems. Processes that better recognize this responsibility, and that provide clear and respected channels for raising concerns, were seen as essential to improving outcomes for both the committee and users.

2.3 Challenges participating in Evolution and Wording work

Implementers described significant difficulty staying engaged across both evolution and wording work. The volume of proposals, number of parallel study groups, and pace of discussion already place heavy demands on limited implementer time. Running EWG/LEWG and CWG/LWG in parallel further amplifies this challenge, making it difficult for implementers to contribute meaningfully to early design discussions while also participating in detailed semantic work.

As a result, implementation concerns may surface only after designs have advanced, when revisiting decisions becomes costly and disruptive. Conversely, design discussions may proceed without timely input on semantic constraints that would later need to be addressed in the wording groups, increasing the risk of rework and misalignment between groups.

2.4 Alignment between committee priorities, user needs, and management expectations

Participants discussed a growing gap between the features the committee is motivated to standardize and the capabilities that users and organizations are able or willing to adopt. While the committee continues to advance new language and library features, many users remain on older standards such as C++17.

Implementers noted that management decisions strongly influence what ultimately gets implemented and deployed. Management typically prioritizes stability, portability, performance, and clear user value, and is less inclined to fund work on features that primarily benefit a small subset of power users or that increase complexity without clear adoption demand. As a result, even committee-approved features may not be implemented if they do not align with organizational priorities or user needs.

Additionally, while any given proposal may fit a customer base the author has considered, the implementers may serve a variety of customer bases with different needs and priorities.

2.5 New features vs bug fixes, and portability

Participants emphasized the importance of balancing new feature development with sustained focus on bug fixes, conformance, and consolidation. New features take time away from addressing existing defects and unresolved core issues while often simultaneously adding new defects to the standard. All these unresolved issues lead to implementations that diverge in behavior and interpretation. This divergence directly impacts portability, as code that works on one implementation may behave differently or fail on another.

When features accumulate faster than they can be fully implemented, integrated, and adjusted, they stack on top of incomplete or inconsistent foundations, further increasing the risk of non-portable code.

Several participants suggested that dedicating more committee effort to defect resolution and high-value fixes would lead to more predictable and interoperable C++ code across platforms and implementations, and ease the adoption of existing features.

3. Suggested actions

3.1 Make cost and trade-offs more visible

We would like the committee to start exploring mechanisms to make cost, resource, and technical debt impact more explicit, or to reason about an overall cost budget per release. We should understand that putting features in the standard is not the end, and be more realistic about the resources we have available.

3.2 Integrate implementer feedback earlier

We would like to ask the committee to start exploring ways of gathering implementer feedback much earlier in the process. One way of doing that would be to create an advisory implementer study group. Such a group could provide mandatory feedback to every proposal brought forward. Some examples of an information that an implementer study group could provide are:

- Is this feature feasible in the way it is specified ?
- What is the cost of this feature for a given implementation?
- Is there any desire from any given customer base/management in this feature?
- Feedback on the reference implementation

It would be useful to have such a group operate on an ongoing basis, for example through GitHub issues, so as to enable implementers who lack the time or resources to attend committee meetings to contribute their perspectives.
Another benefit of having such a study group is that it would foster more collaboration between implementers, which was another topic raised during the meeting.

3.3 Adjust pacing and release focus

We would like the committee to consider ways of slowing down the addition of features into the standard to allow implementers to catch up, and to allow the existing features to improve in quality.

The committee should consider longer standardization cycles or alternating feature-focused and consolidation-focused releases.

We should also explicitly prioritize defect resolution, conformance, and portability work alongside new features.

3.4 Reduce scheduling and participation conflicts

We would like the committee to consider not running evolution and wording groups in parallel where possible. This would allow people with detailed knowledge to be present during design decisions so they can offer feedback early. It may also incentivise people who normally only participate during design decisions to spend more time in the wording groups, which could result in increasing wording knowledge in the committee. Additionally, such a setup would improve the coherence of focus within the committee on any specific feature.

4 Summary

As a committee, we have a shared goal: maintaining a standard that delivers real value to users while remaining implementable, performant, and portable. Our intention is to start a series of conversations about constructive steps we can take as a group toward narrowing the gap between standardization and implementation. We would like to see discussion about improving early communication, making costs and trade-offs more visible, and adjusting pacing and priorities between adopting further features and maturing previously adopted features.