The Beman Project:

Bringing Standard Libraries to the Next Level

DAVID SANKEL

The Beman Project

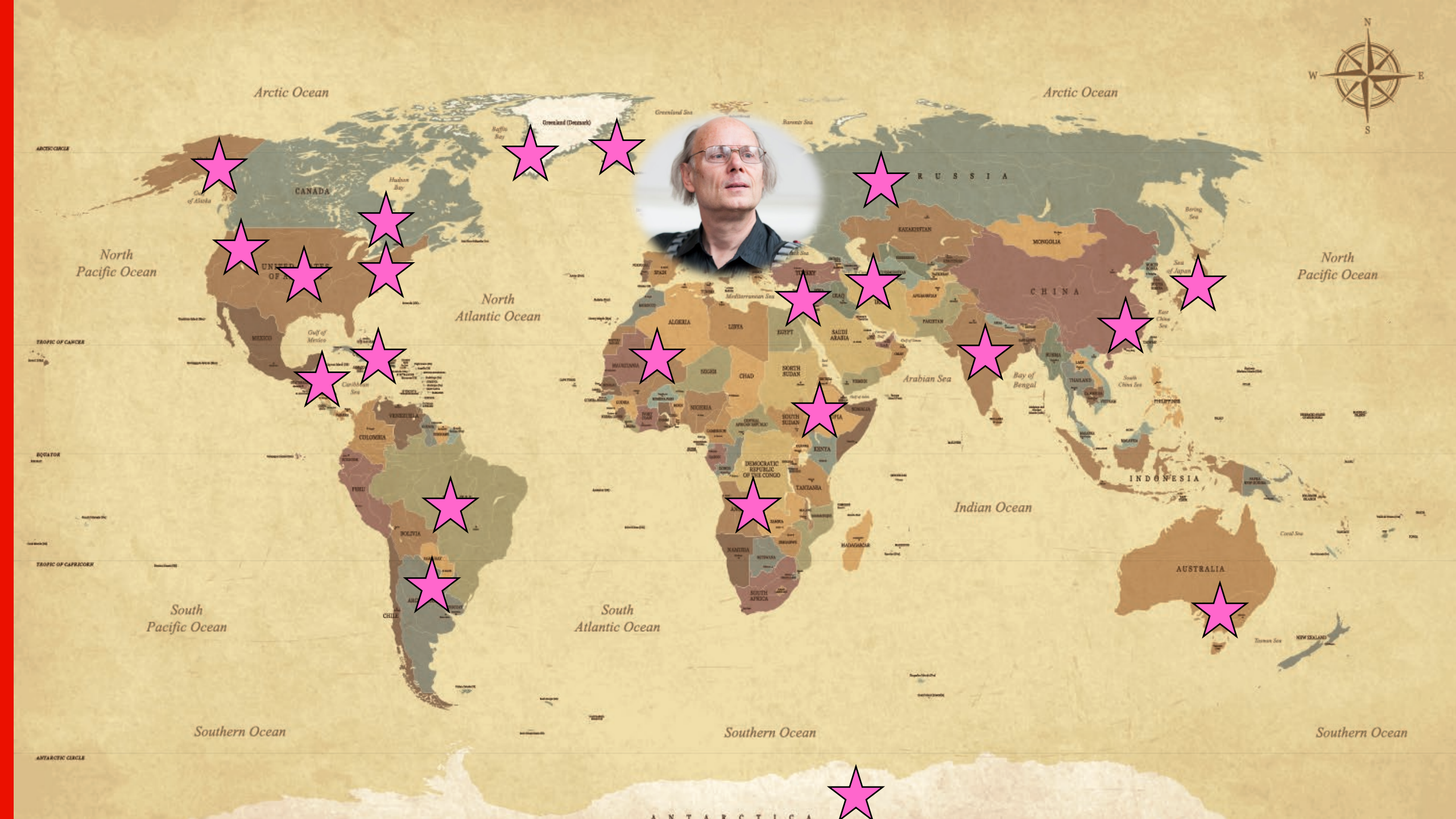
Bringing Standard Libraries to the Next Level

David Sankel | Principal Scientist
Software Technology Lab

Adobe

Image generated with Adobe Firefly

Where does the standard come from?


Arctic Ocean
Arctic Ocean
Greenland (Denmark)
CANADA
RUSSIA
North Pacific Ocean
North Atlantic Ocean
North Pacific Ocean
CHINA
MONGOLIA
KAZAKHSTAN
Arabian Sea
Bay of Bengal
INDONESIA
Indian Ocean
AUSTRALIA
South Pacific Ocean
South Atlantic Ocean
Southern Ocean
Southern Ocean
Southern Ocean
ANTARCTIC CIRCLE
TROPIC OF CANCER
EQUATOR
TROPIC OF CAPRICORN
N
W
E
S

Bristol

Bristol


ISO/IEC
14882:1998

How it really works

- Companies or individuals join the committee at their own election
- It usually costs a bit of money
- Most of the committee consists of regular Jane programmers
  - …with an atypical passion for C++
- Decisions are made via consensus
- Meetings aren’t held on gigantic snowy mountains
  - …it’s a meeting room in a hotel

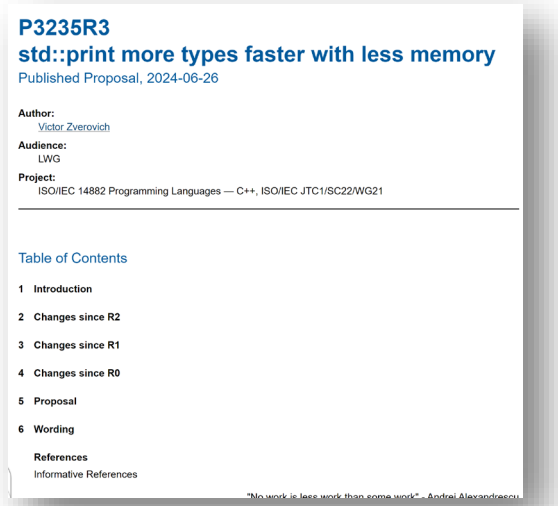
Standard library proposal lifetime

Study group (optional) → Library evolution incubator → Library evolution → Library → Plenary

- Domain specific review
- Prepare proposal for library evolution review

  Determine if sufficient interest to move to next phase
- Design review
- Wording review
- Vote into working draft

  Opportunity to raise last minute concerns.



P3235R3

std::print more types faster with less memory

Published Proposal, 2024-06-26

Author:
Victor Zverovich

Audience:
LWG

Project:
ISO/IEC 14882 Programming Languages — C++, ISO/IEC JTC1/SC22/WG21

Table of Contents

1 Introduction

2 Changes since R2

3 Changes since R1

4 Changes since R0

5 Proposal

6 Wording

References

Informative References

"No work is less work than some work" - Andrei Alexandrescu

Adobe

Adobe's process

- Concept
- Prototype
- Design
- Vetting
- Pre-release
- User studies
- Beta
- General availability

Zach Laine

Jeff Garland

?

Bret Brown

?

?

IL: Meta discussion - proposed improvements to the process (not a policy!):

1. Main benefits I see in this form of project review (Thank you to Jonathan, Anthony and others, this is a great example):
    1. Concise issues on GitHub …
    2. With code Examples…
    3. Which allows discussion history…
    4. And PRs!!! 😊

?

?

?

Inbal Levi

The Boost Foundation

C++Now 2024

David Sankel

Richard Powell

“ Support the efficient design and adoption of the highest quality C++ standard libraries”

– Beman project mission

Adobe

David Sankel

Richard Powell

“ Support the efficient design and adoption of the highest quality C++ standard libraries through implementation experience”

– Beman project mission

Adobe


David Sankel


Richard Powell

“ Support the efficient design and adoption of the highest quality C++ standard libraries through implementation experience, user feedback”

– Beman project mission


David Sankel


Richard Powell

“ Support the efficient design and adoption of the highest quality C++ standard libraries through implementation experience, user feedback, and technical expertise.”

– Beman project mission


Chandler Carruth

Adobe

Beman Dawes (1939-2020)

Boost co-founder

WG21 Library Working Group Chair

"A living demonstration that collegial respect and kindness are fundamental to getting great results." – Dave Abrahams

"He was a friend. He was a giant. He will be missed." – Howard Hinnant

"He was always kind, and through my experience with him I am honored to have had a chance to work with this brilliant member of our community." – Emil Dotchevski

Evolution of the dragon duck

Inbal Levi

Saksham Sharma

Adobe

Beman core principles

1. **Highest quality**. Standards track libraries impact countless engineers and, consequently, should be of the highest quality.
2. **Production-ready**. Production feedback necessitates reliable, well-documented software.
3. **Industry standard technology**. Where there's industry consensus on best practices, we should take advantage. Innovation in tooling and style is not our purpose.
4. **Welcoming and inclusive community**. Broad, useful feedback requires an unobstructed path for using, reviewing, and contributing to Beman libraries. This principle encompasses ergonomics, cross-industry participation, and cultural accommodation.

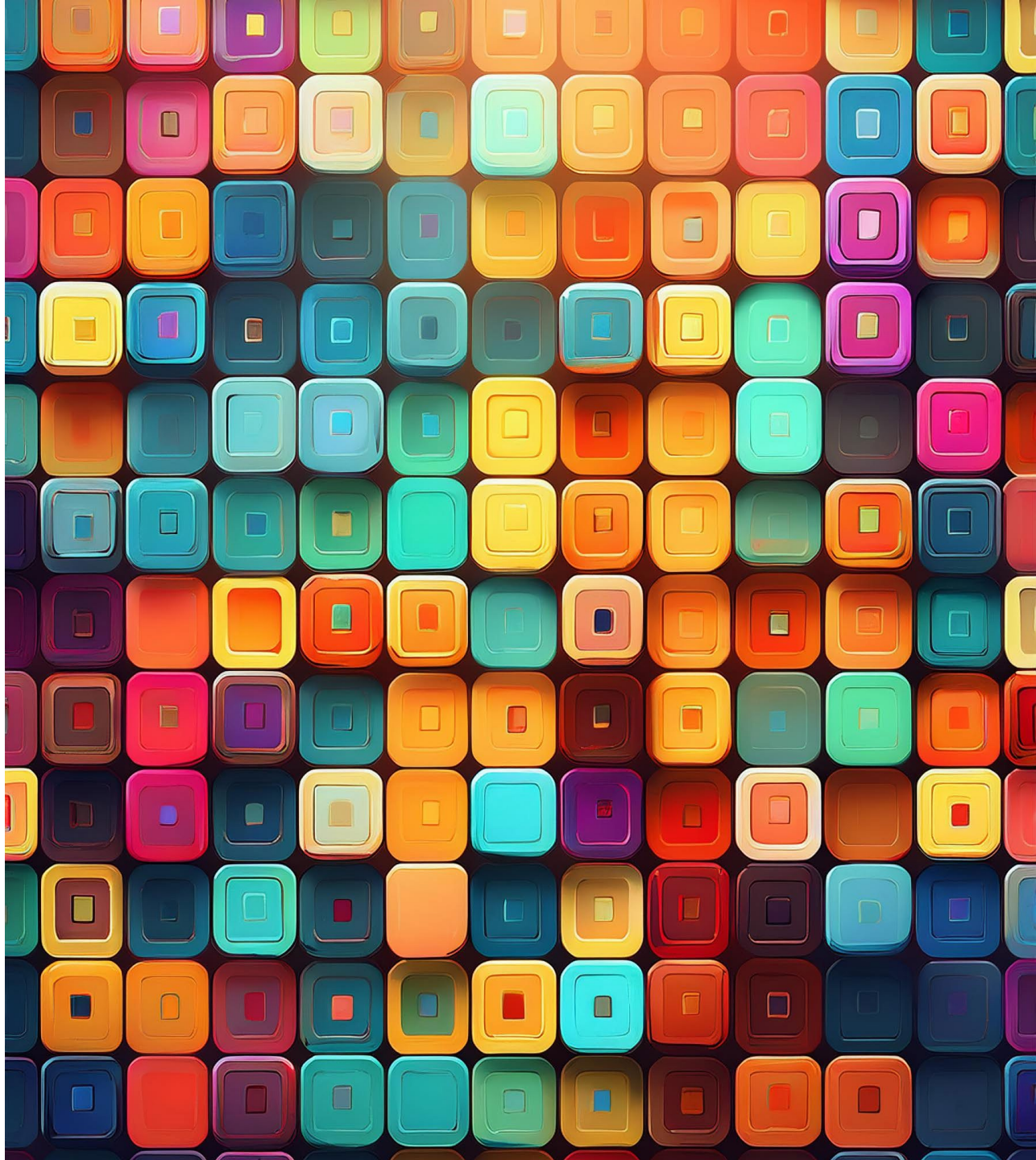
Key Beman standard entries

- Apache License v2.0 with LLVM Exceptions is recommended, but the Boost and MIT licenses are permitted.
- Naming conventions/directory layout (beman.library_name, #include<beman/library_name/foo.hpp>, etc.)
- CMake as build system with several related recommendations

Inbal Levi

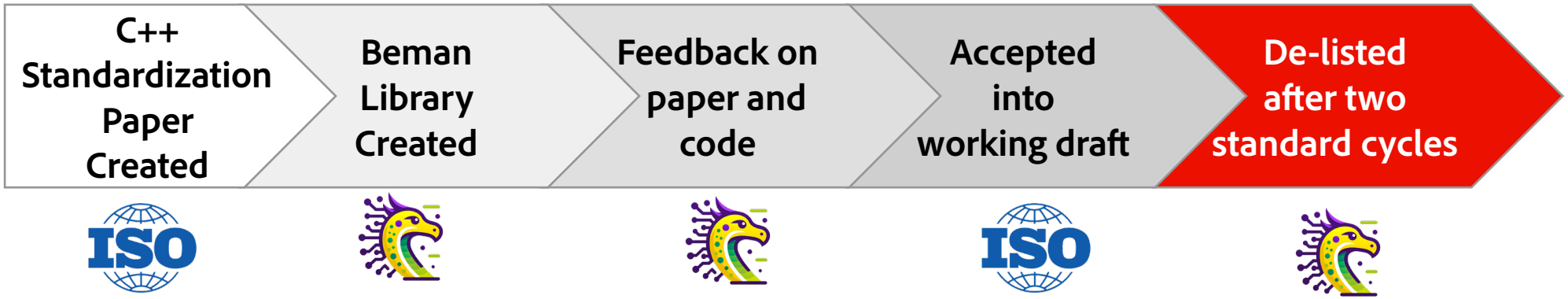
Zach Laine

Bret Brown



Lifetime of a Beman library

Zach Laine



- End of life library removal
  - Encourages std:: adoption
  - Avoids legacy accumulation
- ISO rejected proposals are removed from the official Beman library list

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
#include <beman/optional26/optional.hpp>

namespace opt = beman::optional26;

int main() {

    opt::optional<int> x = /*...*/;

    for (auto i : x) {
        std::cout << i << '\n';
    }

}
```

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
#include <beman/optional26/optional.hpp>

namespace opt = beman::optional26;

int main() {

    opt::optional<int> x = /*...*/;

    for (auto i : x) {
        std::cout << i << '\n';
    }

}
```

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
#include <beman/optional26/optional.hpp>

namespace opt = beman::optional26;

int main() {

  opt::optional<int> x = /*...*/;

  for (auto i : x) {
    std::cout << i << '\n';
  }

}
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
#include <beman/optional26/optional.hpp>

namespace opt = beman::optional26;

int main() {

   opt::optional<int> x = /*...*/;

   for (auto i : x) {
      std::cout << i << '\n';
   }

}
```

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
// A person's attributes (e.g., eye color). All attributes are optional.
class Person {
   /* ... */
public:
   opt::optional<string> eye_color() const;
};

vector<Person> people = /* ... */;

// Compute eye colors of 'people'.
vector<string> eye_colors = people
  | views::transform(&Person::eye_color)
  | views::join
  | ranges::to<set>()
  | ranges::to<vector>();
```

Adobe

Highlights | beman.optional26

Steve Downey

Darius Neațu

```
opt::optional<int&> find_value(std::vector<int> &v, int value) { /*...*/ }
```

Highlights | beman.inplace_vector

Tamás Hadházy

```
#include <beman/inplace_vector/inplace_vector.hpp>

namespace inp = beman::inplace_vector;

//...

inp::inplace_vector<int, 3> v;
v.push_back(1);
v.push_back(2);
v.push_back(3);
v.push_back(4); // throws std::bad_alloc
```

Adobe

Highlights | beman.inplace_vector

Tamás Hadházy

```
#include <beman/inplace_vector/inplace_vector.hpp>

namespace inp = beman::inplace_vector;

//...

inp::inplace_vector<int, 3> v;
v.push_back(1);
v.push_back(2);
v.push_back(3);
v.push_back(4); // throws std::bad_alloc
```

Adobe

Highlights | beman.inplace_vector

Tamás Hadházy

```
#include <beman/inplace_vector/inplace_vector.hpp>

namespace inp = beman::inplace_vector;

//...

inp::inplace_vector<int, 3> v;
v.push_back(1);
v.push_back(2);
v.push_back(3);
v.push_back(4); // throws std::bad_alloc
```

Adobe

Highlights | beman.inplace_vector

Tamás Hadházy

```
#include <beman/inplace_vector/inplace_vector.hpp>

namespace inp = beman::inplace_vector;

//...

inp::inplace_vector<int, 3> v;
v.push_back(1);
v.push_back(2);
v.push_back(3);
v.push_back(4); // throws std::bad_alloc
```

Adobe

Highlights | beman.example

Bret Brown

Darius Neațu

- Exemplar for writing a Beman library
- Best practices for making an Open Source C++ library in general
  - Unit testing
  - Examples
  - Documentation
  - Build
  - Sanitizers
  - Continuous Integration

beman.example: A Beman Library Example

Continuous Integration Tests passing

`beman.example` is a minimal C++ library conforming to The Beman Standard. This can be used as a template for those intending to write Beman libraries. It may also find use as a minimal and modern C++ project structure.

Implements: `std::identity` proposed in Standard Library Concepts (P0898R3).

Usage

`std::identity` is a function object type whose `operator()` returns its argument unchanged. `std::identity` serves as the default projection in constrained algorithms. Its direct usage is usually not needed.

Usage: default projection in constrained algorithms

The following code snippet illustrates how we can achieve a default projection using `beman::example::identity`:

```
#include <beman/example/identity.hpp>

// Class with a pair of values.
struct Pair
{
    int n;
    std::string s;

    // Output the pair in the form {n, s}.
```

Highlights | lit discussion

- In-depth analysis of the lit testing framework
- Demonstrates the kinds of discussions we're having.
  - What is the state of the art?
  - What is the best practice?
  - How and when should new technologies be adopted?

Laura Savino

David Sankel

The Beman Project

Considering Lit for the Beman Project

Date: 2024-07-26
Author: David Sankel

Abstract

The LLVM Integrated Tester (lit) is being proposed as the recommended test framework for Beman Projects. While it supports compilation failure testing, it has limited usage experience, introduces complexity, and has ergonomics challenges. We instead suggest recommending a more popular framework, such as GTest, supplemented with CMake functions adding negative compilation testing capabilities.

Introduction

The Beman Project's mission is to support the efficient design and adoption of the highest quality C++ Standard libraries through implementation experience, user feedback, and technical expertise[1]. Part of this effort includes a set of requirements and recommendations[2] for library authors called the Beman Standard.

At the time of this writing, there is a single testing-related entry which requires tests be executable with CTest:

> **[TOPLEVEL.CMAKE]** REQUIREMENT: There must be a CMakeLists.txt at the repository's root that builds and tests (via. CTest) the library.

While this provides for a consistent user experience, Beman developers could benefit from a test framework recommendation[3].

At the Beman Project's C++Now 2024 kickoff, Louis Dionne recommended adoption of the LLVM Integrated Tester (lit). The benefits of lit are two fold: 1) Lit's negative-compilation tests allow verification of expected compilation failures; and, 2) standard library vendors have shown some

We're starting to see results

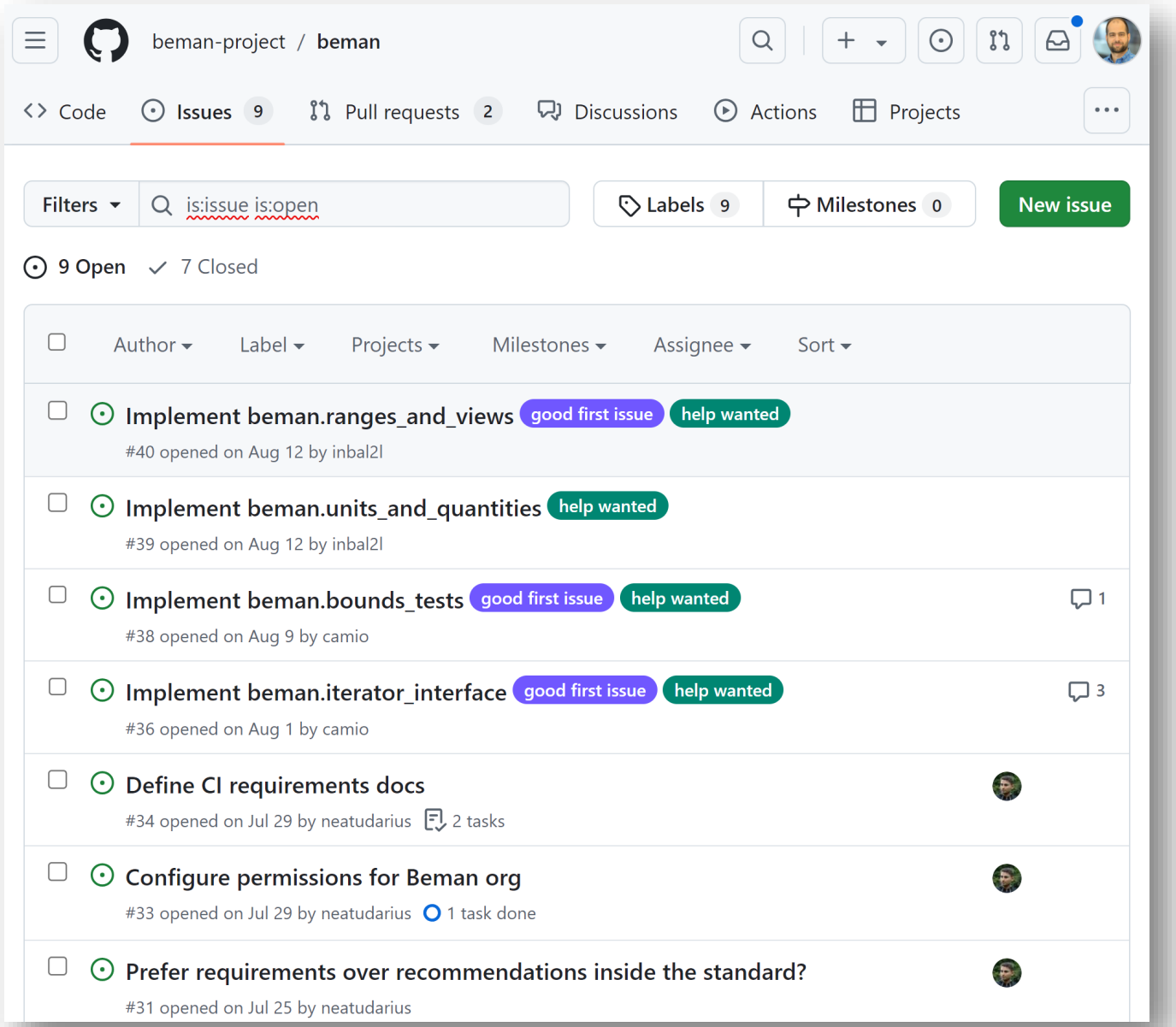
- std::optional
  - Excellent feedback has resulted in substantial paper updates
  - The implementation has been tracking the paper changes
  - "std::optional as a range" got accepted with a full Beman implementation
- std::inplace_vector
  - LWG issues already discovered as part of the implementation and review process
- std::net
  - A much broader feedback base is now available.
  - Pain points identified in std::execution

What's coming?

- Dietmar Kühl's beman.execution26
- Dietmar Kühl's beman.net29
- Alisdair Meredith's beman.testing_resource
- Elias Kosunen's beman.scnlib
- Many others under discussion
- Also…

Your contributions!

- Use the libraries
- Check out our “good first issue” tasks
  - Many have mentors available
- Join the discourse
  - discourse.bemanproject.org
- Join the GitHub conversations
  - Give a code review or comment on one
  - Raise issues (or, even better, PRs)



Who can do this?

camio commented on Aug 1 — Member

This task is to create a new Beman.InplaceVector library implementing Gonzalo Brito Gadeschi et. all's `inplace_vector` paper.

There is a production-quality here licensed under Apache-2.0 WITH LLVM-exception that would be a good start. Another non-production-quality implementation is here.

@gnzblg offered to mentor anyone who wants to take this up.

Hels15 commented on Aug 2 — Member

Hi, @camio. I'd be interested in implementing this for Beman, but first I have to ask: what c++ knowledge is required? Would help from someone who's not professional still be accepted or you are looking for someone with more experience?

camio commented on Aug 2 — Member, Author

Hey @Hels15! For this task, I think intermediate C++ competence would be helpful, but don't worry about asking "beginner" questions. We all start somewhere and are happy to help.

Hels15 commented on Aug 2 — Member

In that case I would be happy to take on the project.

Who's part of Beman?

Leads


Jeff Garland


David Sankel


Inbal Levi

Core contributors


Dietmar Kühl


Steve Downey


Zach Laine


Linus Boehm


Saksham Sharma


Tamás Hadházy


Darius Neațu


Bret Brown


Ben Craig


Richard Powell

Community participants

- Hana Dusíková
- Chandler Carruth
- Frank Miller
- Louis Dionne
- Dave Abrahams
- Sean Parent
- Robert Ramey
- Peter Dimov
- Jonathan Wakely
- Andreas Weis
- And many others…

This week is an awesome time to get started.

Let's talk

discourse.bemanproject.org

github.com/beman-project


Adobe

Attributions

- Photo of Bjarne. By ICPCNews - DSC_5591, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=150572355
- Moses with tablets of the Ten Commandments (1659), painting by Rembrandt. Public Domain